FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání diplomové práce

Název:	Řídicí systém pro model terénního vozítka založený na platformě Arduino
Student:	Bc. Martin Zemánek
Vedoucí:	Ing. Pavel Kubalík, Ph.D.
Studijní program:	Informatika
Obor / specializace:	Návrh a programování vestavných systémů
Katedra:	Katedra číslicového návrhu
Platnost zadání:	do konce letního semestru 2022/2023

Pokyny pro vypracování

1) Prozkoumejte existující možná řešení pro dálkové ovládání vozítek.
2) Pro návrh řízení vozítka vyberte vhodný model obsahující 6 samostatně řiditelných kol.
3) Navrhněte vlastní řešení řízení vozítka s pomocí platformy Arduino za následujících podmínek:
   - každá dvojice kol bude mít vlastní řízení ovládané samostatným Arduinem,
   - řízení bude umožňovat jízdu všemi směry,
   - řízení bude centrálně ovládáno hlavní deskou s Arduinem,
   - veškeré řízení pohybu vozítka bude provedeno pomocí dálkového ovládání,
   - vozítko bude na sobě obsahovat další senzory potřebné pro pohyb a detekci překážek pro všechny směry,
   - vozítko bude umět automaticky měnit maximální rychlost v závislosti na vzdálenosti od překážky, a ve všech směrech možného pohybu.
4) Počítejte s možným pozdějším rozšířením o platformu Raspberry PI umožňující použití například rozhraní wifi, SD karty a přídavné kamery.
5) Navržené řešení zrealizujte a řádně otestujte.

Elektronicky schválil/a prof. Ing. Hana Kubátová, CSc. dne 2. února 2022 v Praze.

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Diplomová práce

Řídicí systém pro model terénního vozítka založený na platformě Arduino

Bc. Martin Zemánek

Katedra číslicového návrhu
Vedoucí práce: Ing. Pavel Kubalík, Ph.D

28. prosince 2022

Poděkování

Rád bych na tomto místě poděkoval Ing. Pavlu Kubalíkovi, Ph.D, za jeho trpělivost a spolupráci při tvorbě této práce. Rád bych poděkoval Bc. Michalu Švecovi, za zapůjčení 3D tiskárny. Velký dík patří rodině a kamarádům, kteří mě motivovali.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo"), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 28. prosince 2022 .....

České vysoké učení technické v Praze
Fakulta informačních technologií

Odkaz na tuto práci

Zemánek, Martin. *Řídicí systém pro model terénního vozítka založený na platformě Arduino*. Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2022.

Abstrakt

Práce se zabývá tvorbou řídícího systému pro předem vytvořený podvozek, který je založen na platformě Arduino. Tento řídící systém umí přizpůsobit svůj chod informacím obdrženým z okolí. Následně tento systém umí informace o svém stavu sdílet s nadřazeným systémem. Tento systém je možné ovládat za pomocí rádia či za pomocí výstupu z nadřazeného systému.

Klíčová slova Arduino, Řídící systém, H-můstek

Abstract

In this thesis the main focus of the work is a controling system for pre-made undercarrige. This system is based on Arduino platform. The system is able to adapt it's behaviour based on informations from it's surrounding. This system is also able to share these informations with superior system. This system is controllable by a radio transmitter or by commands from the superior system.

Keywords Arduino, Control system, H-bridge

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Řídící systémy jsou dnes součástí běžného života každého z nás. Od nejjednodušších domácí spotřebičů, přes zábavní elektroniku, až po automobily a komplexní systémy využívané ke specifickým účelům. Tyto systémy usnadňují život v naší civilizaci a umožňují dříve neproveditelné skutky uskutečňovat s lehkostí. Každý z těchto systémů tvoří celek, plnící svůj účel. Tyto celky pak mohou být užity jako komponenty většího systému.

Náplní této práce bylo navrhnout a zrealizovat menší řídící systém. Účelem je řízení šestikolového podvozku. Jakožto mobilní systém vyžaduje určitou míru samostatnosti a musí předpokládat, že jeho uživatel nemá plné vědomí o každé jednotlivé komponentě. Systém proto musí zpracovávat vstupy ze svých periferií a na jejich základě upravit chování vozítka.

Vzhledem k jednoduchému základu jsem chtěl využít platformu Arduino[15]. Tato platforma, oproti jiným platformám, je snadno využitelná a lehce dostupná. Nicméně je vhodnější pro jednodušší projekty, proto jsem chtěl prozkoumat možnosti rozdělení na menší podsystémy a jejich propojení za pomocí sběrnice CAN[16], která se k tomuto účelu užívá.

Kapitola 1

Cíl práce

Cílem této práce je prozkoumat existující možná řešení pro dálkové ovládání vozítek, navrhnout a zrealizovat řídící systém pro šestikolový podvozek, který je založen na platformě Arduino[15].

Hlavní funkcionalitou systému je ovládání vozítka na základě přijatých povelů z RC ovládání. Vozítko musí být vybaveno senzory pro detekci překážek ve všech možných směrech pohybu. Na základě výstupu senzorů bude vozítko schopno omezovat svoji maximální rychlost.

Každá dvojice kol by měla být ovládaná samostatným Arduinem[15]. Každý z těchto modulů by měl být řízený za pomocí centrálního Arduina[15]. Vzniklé řízení musí umožňovat jízdu vozítka všemi směry.

Následně má být vozítko připraveno na připojení rozšiřujícího modulu založeném na platformě Raspberry Pi [17]. Tento modul by měl být schopen vozítko ovládat a vyčíst jeho stav.

Zařízení je třeba zpracovat způsobem takovým, aby bylo využitelné pro výukové účely.

Na konec je třeba vytvořené řešení otestovat.

Kapitola 2

Analýza a návrh

V této kapitole se zabývám několika kroky. Prvním krokem je zkoumání existujících řešení. Prozkoumal jsem čtyři řešení, která byla dostatečná k navržení vlastního řešení na zvolené platformě.

Druhým krokem byla analýza. V tomto kroku jsem rozložil problém na jednotlivé podproblémy, které vzešly ze zkoumání existujících řešení.

Následujícím krokem byla volba použité technologie. Tato část je rozdělena na dvě sekce. První sekcí je volba součástek. Druhou sekcí je volba software.

Posledním krokem je samotný návrh. V tomto kroku provádím návrh systému, jako celku. Jedná se o rozložení součástek, jejich rozdělení na skupiny a funkce jednotlivých skupin.

2.1 Rešerše

Zkoumal jsem čtyři řešení. Následně rozebírám použité technologie a zvolené postupy. Základem všech řešení je ' centrální řídící modul řídící vozítko. Od této části se odvozují požité technologie a postupy.

Dalšími podproblémem jsou řízené parametry. Ty určují, jaké parametry systém nastavuje a jak rychlá vůbec může být odezva systému. Nejvíce mě zajímá řízení motorů.

Významnou součástí jsou pak měřené veličiny. Řídící systém musí být schopen se přizpůsobit reálnému okolí a svému stavu. Tyto informace může odvodit pouze za pomocí měření veličin.

Posledním zkoumaným podproblémem je komunikace s uživatelem. Nejprve je třeba přijmout povely od uživatele, které vozítko zpracuje. Následně musí vozítko poslat informace o svém stavu a o stavu okolí zpět uživateli.

2.1.1 Mobilní robot schopný pohybu ve venkovním prostředí[1]

Toto řešení je popsáno v diplomové práci, která vznikla na Fakultě Informatiky na Českém Vysokém Učení Technickém v Praze. Cílem této práce bylo realizovat robota schopného pohybu ve venkovním prostředí.

Řídící systém tohoto řešení je založen na modulech STM32 Nucleo-64[18], které jsou založeny na mikrokontrolerech STM32F401RET6[19]. Systém dále používá tyto moduly dvěma způsoby.

První je modul pro ovládání jednotlivých motorů. Tento modul ovládá přímo dvojici motorů. Je na něj připojena zpětná vazba z těchto motorů.

K ovládání motorů využívá H-můstků. Konkrétně využívá implementace VNH3SP30[7]. Toto řešení umožňuje přesné jednoduché ovládání motorů (tj. změna směru, otáčení, apod.). Tento H-můstek je také využit k monitorování jejich stavu.

Zpětná vazba je realizována pomocí Enkodéru. Enkodéry jsou realizovány za pomocí Hallových senzorů. Pomocí těchto senzorů se v tomto řešení kontroluje rychlost a směr pohybu každého kola.

Kombinace enkodéru a H-můstku umožňuje přesnou kontrolu polohy a rychlosti otáčení každého jednotlivého kola.

Každý z těchto modulů je poté připojen k řídícímu modulu. Tato komunikace je realizována za pomocí SPI linky[20]. Moduly mají na starost pouze udržení konkrétní rychlosti a směru otáčení každého z motorů.

Dále tento systém obsahuje řídící modul. Tento modul zpracovává vstupy ze senzorů a dálkového ovládání. Řídící modul umožňuje připojení Raspberry Pi[17], či jiného alternativního ovládání, pomocí rozhraní USART[21]. Forma příkazů, pomocí kterých by se robot ovládal, není v práci popsána.

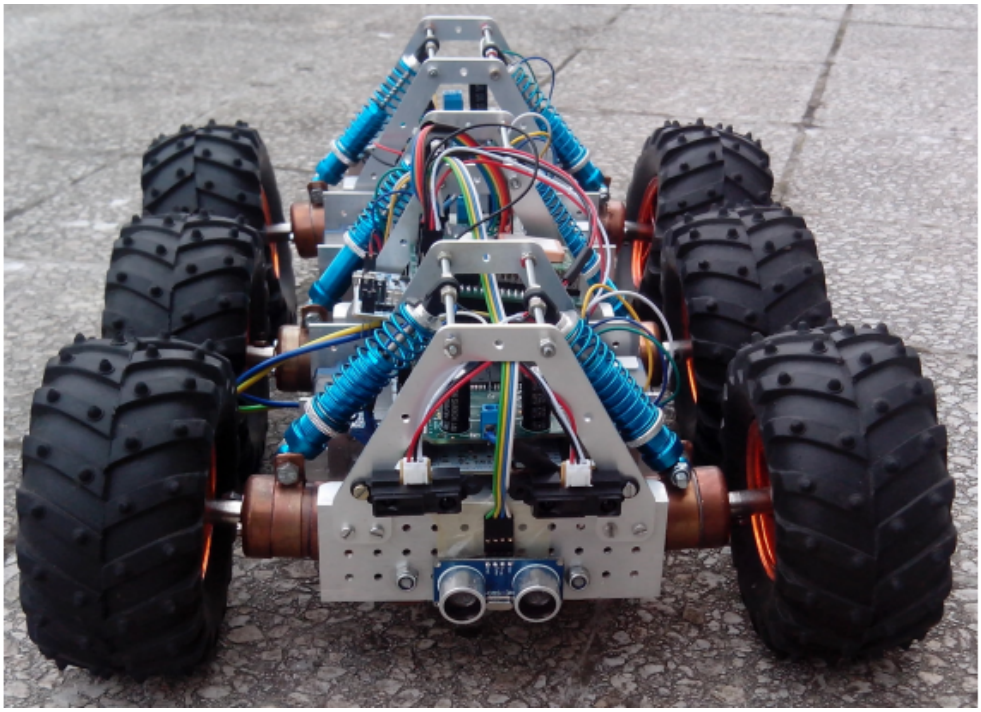
Modul umožňuje příjem pokynů od uživatele. Ke zpracování výstupu ovladače je využit přijímač Spektrum AR400[12]. Tento modul umožňuje příjem pokynů z několika RC ovladačů. Řešení pracuje konkrétně s ovladačem Spektrum DX5e[22]. Na řídící modul jsou zapojeny senzory pro vnímání okolí.

Pro vnímání bezprostředního okolí zařízení využívá ultrazvukový HC-SR04 [23] a IR senzor GP2Y0A21[24]. Pro snímání náklonu, směru a akcelerace je využit senzor GY-80[25]. Tento modu je kombinací tří-osového gyroskopu, tří-osového akcelerometru a tří-osového magnetometru. Tento senzor zajišťuje kontrolu správného pohybu robota. Pro vnímání polohy robota je také použit GPS modul Trimble Anapala ACM.

Vozidlo má dále senzor pro měření stavu baterie. Pro detekci stavu baterie využívá práce AD převodníku, který je součástí řídící jednotky.

Dále se modul zabývá měřením teploty a vlhkosti. K tomuto je využit senzor SH15[26].

V tomto případě se jedná o rozšiřující funkcionalitu, neboť vlhkost ani teplota nemá vliv na řízení robota [1].



Obrázek 2.1: Mobilní robot schopný pohybu ve venkovním prostředí[1][5]

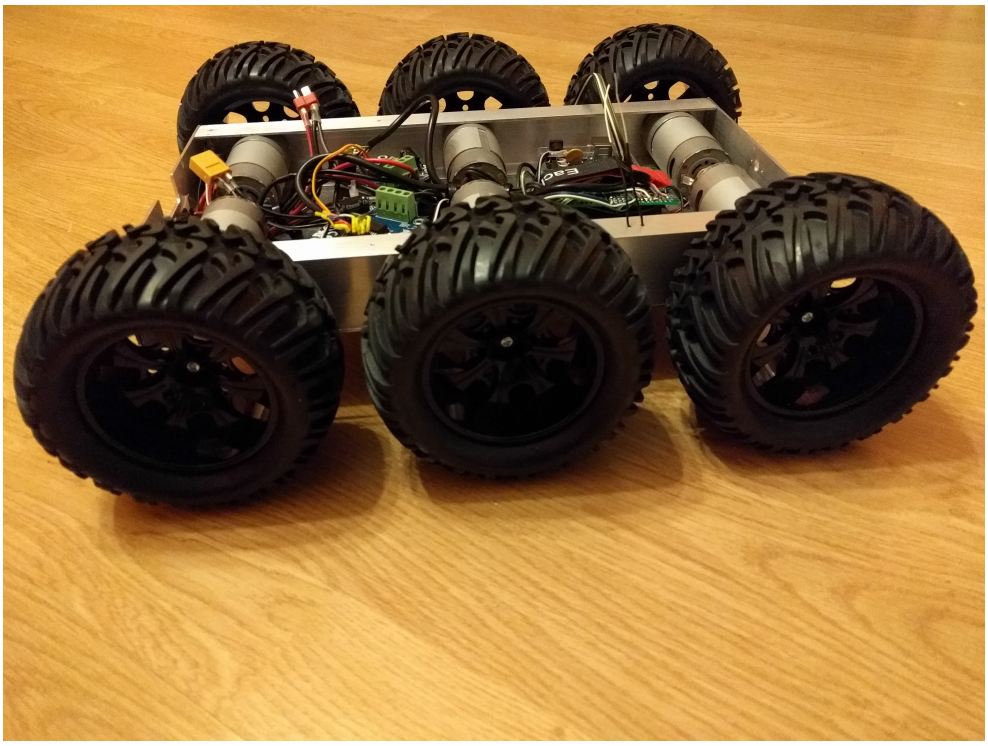
2.1.2 Remote Controlled 6WD All Terrain Robot[2]

V této práci se jedná o šestikolový podvozek. Projekt vznikl za účelem postavit vozítko, které uveze těžší náklady. Zajišťuje pouze přijímání povelů od uživatele a ovládání skupin motorů.

Řídícím modulem projektu je Arduino MEGA 2560[10], který je jediným řídícím modulem v tomto projektu. Řídící modul ovládá dvě trojice kol za použití H-můstků VHN2SP20[27]. K napájení používá dvou akumulátorů, a to 11 [V] akumulátor k pohonu motorů a 7.4 [V] k napájení řídícího modulu. K přijímání signálu od uživatele je použit radiový vysílač a přijímač. Řídící modul je poté zapojen na nadřazený modul, který umožňuje vnímat okolí a přenášet obraz. Podvozek používá k pohybu diferenciální řízení.

Řídící modul tohoto projektu není vybaven senzory pro monitorování stavu motorů. Motory v tomto projektu není možno ovládat jednotlivě. Projekt také neumožňuje detekci překážek v okolí řídícím modulem.

Řídící modul počítá s připojením na modul, který zajišťuje rozšiřující funkce [2].



Obrázek 2.2: Remote Controlled 6WD All Terrain Robot[2]

2.1.3 6WD-Arduino-robot-car [3]

[3]

Tento projekt je vozítkem založeným na vlastní vytvořené platformě. Jako komunikační modul používá Arduino UNO [9] nebo Raspberry Pi Model 3 B [11]. Tento projekt počítá s ovládáním přes komunikační modul a nepočítá s ovládáním pomocí vysílačky. Dálkové ovládání je realizováno za použití Bluetooth nebo Wi-fi. Vozítko používá k pohybu diferenciální řízení. K ovládání jednotlivých motorů je použit složitý modul s drivery. Arduino je zde pouze v roli komunikačního modulu. Řešení je rozšířeno o řadu senzorů.

Všechno v tomto řešení je realizováno za pomocí komplexního modulu. S tímto modulem je třeba komunikovat. Konkrétní komunikace s uživatelem je řešena pouze pomocí SPI a sériové linky. Toto řešení je snadno použitelné ve složitějším projektu [3].

Obrázek 2.3: 6WD-Arduino-robot-car[3]

2.1.4 Dálkově ovládané čtyřkolové vozítko založené na platformě Arduino[4]

Toto řešení je popsáno v bakalářské práci, která vznikla na Fakultě Informatiky na Českém Vysokém Učení Technickém v Praze. Cílem této práce bylo zkonstruovat čtyřkolové vozítko založené na platformě Arduino.

Řešení se zabývá ovládáním motorů, zpracováním vstupů od uživatele, monitorováním prostředí a úpravy chování podvozku.

Řešení využívá pouze jednoho modulu. Řídící modul je založen na platformě Arduino MEGA 2560[10]. Tato platforma obsahuje mikrokontroler od firmy Microchip ATmega 2560[28]. Toto je jediný řídící systém v celém řešení a ovládá všechny procesy a senzory.

Pro ovládání každé dvojice motorů je zde využit H-můstek, stejně, jako u předchozího řešení. H-můstek je realizován pomocí modulu L298N[6]. Tento H-můstek umožňuje ovládání dvou motorů z jednoho modulu.

Řešení využívá přijímač MK610[29]. Řídící systém nicméně neumožňuje připojení složitějšího řídícího systému. Jiný způsob ovládání vozítka popsaný v bakalářské práci není.

Řešení se zabývá zpracováním vzdálenosti od překážky. K tomuto využívá senzor HC-SR04 [23]. Na základě tohoto senzoru vnikají podmínky omezující rychlost vozítka.

Práce nezkoumá aplikaci složitějších senzorů, ani přesného natavení rychlosti motorů. Jediná zpětná vazba v rámci tohoto řešení je přímý vizuální kontakt uživatele s vozítkem.

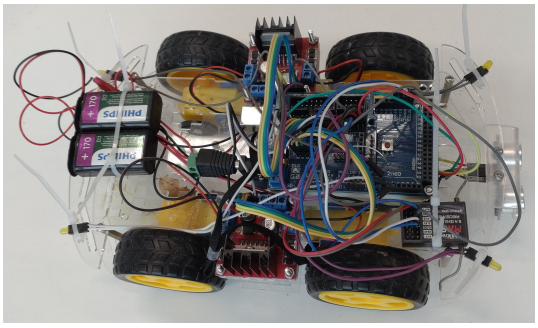
Výhodou řešení je cena. Nejdražší položkou je ovládací modul.

Řešení se nezabývá měřením spolehlivosti vozítka. Dojde-li k poruše, pak jeho řídící systém nemá možnost poruchu odhalit.

Počítá se s využitím běžně dostupných devíti voltových baterií.

K realizaci některých částí vozítka byl použit 3D tisk.

Největší nevýhodou tohoto systému je absence zpětné vazby, krom vizuálního kontaktu[4].



Obrázek 2.4: Dálkově ovládané čtyřkolové vozítko založené na platformě Arduino[4]

2.2 Analýza

Všechna popisovaná řešení potřebovala v první řadě řídit motory. Z motorů je třeba přijímat odezvu, pro kontrolu jejich funkce. Styl ovládání motorů je závislý na typu podvozku.

Následně je vždy třeba přijímat povely od uživatele. Uživatel může komunikovat s vozítkem různými cestami. Pro každou z těchto cest je třeba zpracovat příkazy a zachovat se podle nich.

Vozítko by také mělo být schopno zhodnotit proveditelnost zadaných povelů. Je-li vyhodnoceno, že je povel neproveditelný, je třeba takový povel modifikovat, či jej neuposlechnout.

Vozítko by také mělo svůj stav dát najevo uživateli. Pomocí komunikačního protokoly jsou chyby hlášeny uživateli.

2.2.1 Podvozek

Podvozek je z hlediska navrhovaného řídícího systému ovládaný systém. Od jeho vlastností se odvíjí, jaké parametry a jakým způsobem ho musí řídící systém ovládat. Na podvozku také závisí, jakým parametrům se musí systém přizpůsobit.

2.2.1.1 Ovládání motorů

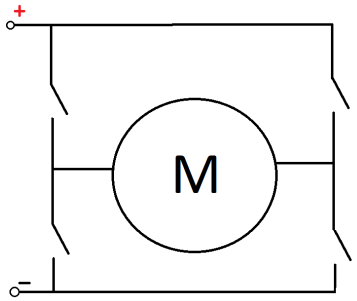
Ovládání motorů je závislé na několika faktorech. Rozhodujícím faktorem je typ ovládaného motoru. Z tohoto vychází ovládané veličiny, způsob měření veličin a napájení motoru.

Vozítko, jaké navrhuji, vyžaduje napájení z akumulátoru. Baterie jsou zdrojem DC proudu. Proto je vhodné zvolit DC motor. Směr otáčení DC motoru je pak možno ovládat změnou směru proudu. Často užívaným prvkem, který se k tomuto účelu používá, je H-můstek. Ten umožňuje modifikovat sílu použitou k otáčení motoru. Na Arduinu je za tímto účelem použita PWM modulace. Ta umožňuje nastavit, aby motor zabíral jen po určité procento času, čímž dojde k efektivnímu omezení jeho výkonu a tudíž i rychlosti.

2.2.1.2 Princip zatáčení

Existuje několik možných typů zatáčení vozítka. Nejběžnější jsou dva způsoby. Jiné typy řízení podvozku jsou pak většinou kombinací těchto dvou.

Prvním způsobem je **Ackermanovo řízení**. Tento způsob se běžně užívá v autech. Změnu směru provádíme změnou v ose otáčení náprav. Tento styl vyžaduje dva způsoby ovládání motorů a zpětné vazby. Prvním způsobem je pouze ovládání hnacích motorů vpřed a vzad. Druhým způsobem je přesné ovládání úhlu změny u řídících náprav. U první části je tedy třeba řídit pouze výkon, kdežto u druhé je třeba přesně odečíst úhel, o který se náprava otočila. [30]



Obrázek 2.5: H-můstek

Druhým způsobem je **Diferenciální řízení**. S tímto typem řízení se běžně setkáváme u pásových vozidel. Na rozdíl od Ackermanova stylu řízení potřebuje pouze jeden způsob řízení motorů. U těchto motorů je pak třeba řídit pouze rychlost v daný moment. Rozdílem výkonu, mezi jednotlivými stranami vozítka pak dostaneme zatočení. Na rozdíl od Ackermanova stylu řízení také umožňuje diferenciální styl řízení otáčení na místě. [31]

Pro svoji práci jsem zvolil podvozek, který je stavěn pro **diferenciální řízení**. Toto umožňuje přesnější ovládání polohy vozítka a snadnější ovládání z pozice řídícího systému.

2.2.1.3 Motory

Pro **diferenciální řízení** je třeba ovládat výkon každého jednotlivého kola. Za tímto účelem je vhodné použít malé motory, kde každý z motorů ovládá přesně jedno kolo. Motory dále, jako celek musí mít dostatečnou sílu, aby byly schopny operací s celým vozítkem dle zadaných parametrů.

Jak jsem se již zmínil vozítko bude napájeno z akumulátoru. K napájení z baterie jsou vhodné DC motory, protože výstup akumulátoru je také DC.

Na podvozku, který jsem si zvolil, byly již nasazeny DC motory, nicméně část z nich byla poškozena způsobem, který vylučoval jejich další užití v projektu. Bylo proto třeba najít jejich náhradu. Výběr náhrady je popsán v kapitole Volba Součástek.

2.2.1.4 Zpětná vazba

Z motorů je možné získávat zpětnou vazbu a také je možno měřit několik různých veličin. V ideálním případě je motor zvolený pro řešení již vybaven senzory pro zpětnou vazbu.

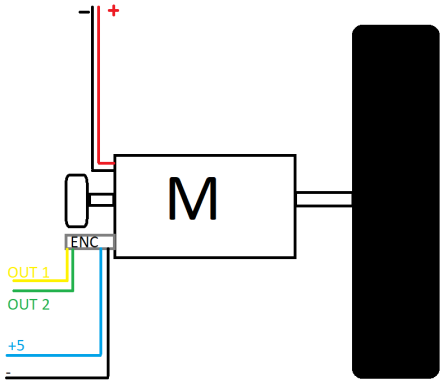
Zpětnou vazbu z motoru je možno získat měřením otáček. K tomuto účelu se používají enkodéry. Pro realizaci enkodéru je třeba rozdělit úhel otočení

motoru na pevně dané úseky. Poté je třeba přidat značku na každý z těchto úseků, kterou je možné přečíst za pomocí řídícího systému. Tyto značky musí být připevněny přímo k rotoru motoru. Pro detekci směru je poté třeba mít dvě čidla, která jsou o půl fáze posunuta.

K realizaci je možné použít Hallovy senzory. Tyto senzory snímají magnetické pole. Řešení vyžaduje, aby k rotoru motoru byl připojen magnet. Halovo čidlo pak spíná výstup v závislosti na své pozici v magnetickém poli.

Další možností realizace enkodéru je laserová závora a disk s průhlednými a neprůhlednými úseky. Tento disk pak přerušuje závoru v závislosti na poloze rotoru.

Pro realizaci ve svém projektu jsem se rozhodl použít Hallovo čidlo. Hlavním důvodem je to, že je možné pořídit DC motory vybavené tímto čidlem. Dále je toto řešení podstatně menší oproti laserovým čidlům.



Obrázek 2.6: Motor s enkodérem

2.2.2 Řídící modul

Řídící modul představuje jednotku, která ze zadaných vstupů vytvoří žádané výstupy. V tomto modulu musí být implementována základní logika vozítka.

Nejjednodušším způsobem je přímo napojit přijímací modul z vysílačky na motory a řídit je přímo. Toto řešení neumožňuje samostatnost vozítka a neumožňuje uživateli získat informace o vnitřních stavech. Vozítko musí být pod přímým dohledem uživatele. Toto řešení je velmi levné a RC přijímače jsou k tomuto účelu přímo stavěné.

Dalším řešením bylo zařadit mezi přijímač ovládání a motory řídící modul. Řešení tímto způsobem vyžaduje velmi výkoný modul, jinak jsou možnosti tohoto řešení velmi omezené. Při užití menšího podvozku může být toto řešení dostačující. Nicméně při větším množství periférií může docházet k přetížení modulu nebo velkému opoždění jeho reakce.

Nejsložitějším řešením byla realizace za pomoci několika programovatelných modulů. U tohoto řešení je zapotřebí realizovat komunikaci mezi moduly. Výhodou je možnost oddělení jednotek ovládající motory a jednotky ovládající celé vozítko, což umožňuje zpracování většího množství vstupů a přesnější ovládání výstupů. Pro své řešení jsem se rozhodl využít tento postup.

2.2.2.1 Komunikační rozhraní

Řídící modul může mít několik rozhraní. V první řadě je třeba vyřešit komunikaci v rámci komunikačního modulu.

Na projektu *Mobilní robot schopný pohybu ve venkovním prostředí*[1] je k tomuto účelu využita komunikace za pomocí SPI rozhraní. SPI rozhraní realizuje přímou komunikaci mezi dvěma moduly. Jeden z modulů je v roli master, druhý v roli slave. Master poskytuje rozhraní hodiny. Následně také určuje, kdy komunikace probíhá. Pro komunikaci s více zařízeními je třeba buď mít na jednom modulu více SPI linek, nebo je možno použít digitální výstupy nadřazeného modulu k určení, se kterým z podřízených modulů modul zrovna komunikuje. Z toho vyplývá, že chceme-li na tuto linku přidat další modul, je třeba tuto linku modifikovat a velmi změnit hardware a firmware modulů zapojených do komunikace.[20]

Další možností komunikace mezi moduly je použití sériové linky. Běžně se toto rozhraní používá ke komunikaci propojenou 1:1. Z tohoto vyplývá, že pro komunikaci mezi více moduly by bylo třeba, aby master zařízení této komunikace bylo vybaveno stejným počtem rozhraní, jako u SPI linky. Na toto rozhraní je možné také zapojit více zařízení. Zde je potřeba zajistit několik věcí. V datech je třeba rozlišit, komu je odeslaný paket určený, poté je třeba zajistit, aby na lince odesílalo pouze jedno zařízení. Jedná se o velmi komplikovaný problém. [32]

Pro komunikaci více zařízení na jedné lince je možno využít několik různých způsobů. Vhodným rozhraním pro komunikaci více zařízení je sběrnice CAN. Na sběrnici může být několik zařízení. Výběr zprávy, která se odešle, je rozhodnut v závislosti na hlavičce zprávy. Všechny moduly se pokusí odeslat najednou. Je-li modul přehlasován, pak začne pouze poslouchat. Každý modul musí rozhodnout, zda je zpráva určena pro něj. Výhoda této sběrnice spočívá v tom, že rozhodování o přijetí zprávy probíhá jíž na hardwarové úrovni. Příjemce provádí kontrolu korektnosti zprávy za pomocí kontrolního součtu. [16]

Dále je třeba realizovat komunikaci s nadřazeným modulem. Volba tohoto rozhraní závisí na nadřazeném modulu. Není vhodné použít stejné rozhraní, které je použito ke komunikaci mezi primárním a sekundárními moduly. Na tomto rozhraní bude prováděná častá komunikace a je kritické, aby nebyla přerušena. K realizaci tohoto rozhraní je možné použít stejné technologie, jako k realizaci vnitřního rozhraní. Je třeba zvolit rozhraní podporované u obou

komponent. Sériová linka je snadno použitelná k tomuto účelu. Také je možné využití SPI linky pro komunikaci.

2.3 Volba součástek

Na základě analýzy vzniklo několik problémů, pro které bylo třeba najít hardwarové komponenty. V této sekci jsem hledal komponenty použitelné k realizaci těchto podproblémů.

2.3.1 Podvozek

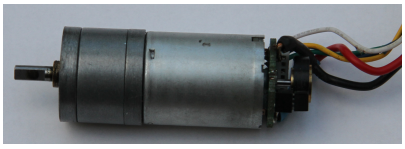
Pro realizaci jsem se rozhodl využít podvozek vytvořený pro projekt *Mobilní robot schopný pohybu ve venkovním prostředí* z roku 2014 [1]. V rámci tohoto projektu byl vytvořen šestikolový podvozek s diferenciálním řízením.

2.3.2 Řízená soustava

Řízená soustava se skládá z motoru a modulu, který umožňuje jeho ovládání. Původní podvozek byl poškozen, tudíž některé jeho části bylo třeba modifikovat, či nahradit.

2.3.2.1 Motory

Na původním podvozku byly nainstalovány motory s enkodérem. Jednalo se o motory napájené 6 [V] s maximálními otáčkami 285 [ot/min]. [33] Bohužel tyto motory byly poškozené a jejich výroba byla přerušena, a proto bylo třeba najít jejich náhradu.



Obrázek 2.7: Motor [1]

Jako náhradu výrobce doporučuje variantu motoru s přidanou krytkou.[34] Motor by byl perfektní náhradou, ale pro použití v dané aplikaci je zbytečně drahý.

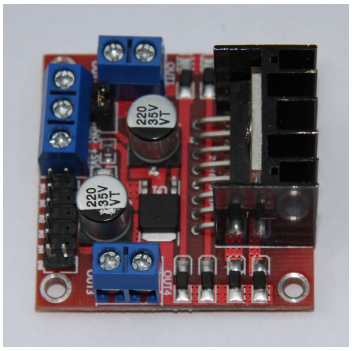
Prozkoumal jsem výrobky jiných výrobců a podařilo se mi najít motory s podobnými parametry. Velkou výhodou u těchto motorů byla cena. Motory mají trochu jinak řešený enkodér. Místo jednoho Hallova čidla, je zde použita dvojice menších senzorů. Výstupy enkodérů jsou shodné s původními. [35]

Jako náhradu jsem proto zvolil levnější variantu.

2.3.2.2 Řízení motorů

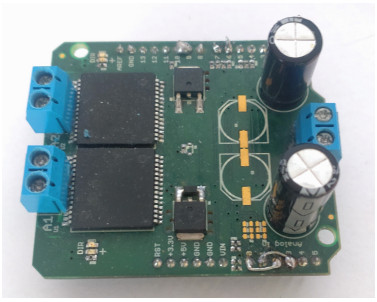
Pro realizaci ovládání motorů jsem se rozhodl využít H-můstky. H-můstek umožňuje jednoduché řízení motorů za pomocí PWM a dvou směrových signálů.

Jako první jsem se rozhodl vyzkoušet H-můstek L298N. Tento můstek byl součástí modulu. Modul jako celek je osazen i 5 [V] zdrojem. K řízení tohoto modulu se používá 5 [V] obvod. Umožňuje řízení 5-35 [V] obvodu. Nevýhodou tohoto modulu je, že je omezen pouze na 2 [A] proudu na kanál. [6]



Obrázek 2.8: Modul L298N[6]

Druhou variantou je H-můstek VNH3SP30 [7]. H-můstek je opět součástí modulu. Umožňuje řízení obvodu se stejnými napětími, jako L298N. Podstatnou výhodou VNH3SP30 je, že ovládaný obvod může mít proud až 30 [A]. Opět je řízen 5 [V] obvodem. Nevýhodou tohoto obvodu je absence zdroje pro řídící obvod, vyžaduje tedy samostatné napájení. Výhodou je, že jako celek se velmi snadno připojí k řídícímu modulu. [7]



Obrázek 2.9: Modul VNH3SP30[7]

Na konec jsem vyzkoušel obě zmíněné varianty. Výsledku tohoto zkoumání se věnuji v pozdějších kapitolách.

2.3.3 Řídící modul

Jak už bylo zmíněno v analýze této práce, pro projekt jsem se rozhodl využít model s několika moduly. Řešení má výhodu v tom, že je možné zpracovávat výstup každého z motorů rychleji a přesněji. Nicméně je pak nutné vytvořit řídící (primární) modul. Tento modul musí zpracovávat výstupy od všech sekundárních modulů a senzorů a poté vyhodnocovat, jaké povely zadat sekundárním modulům.

Sekundární modul musí zpracovávat příkazy přijaté od primárního modulu. Je třeba, aby vyhodnotil zda motory vracejí očekávané výstupy. V případě detekce poruchy systému dá na vědomí tuto skutečnost primárnímu modulu a v takovém případě je nutné, aby ochránil svěřené motory. Sekundární moduly musí být od sebe rozeznatelné v rámci komunikace pro pozdější diagnostiku.

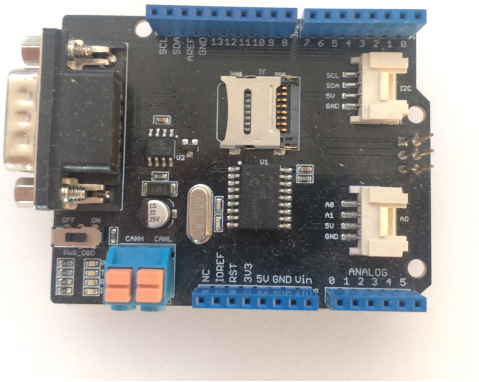
2.3.3.1 Komunikace

Pro komunikaci v rámci řídících modulů v předchozích projektech je zvolena SPI sběrnice. Tato sběrnice realizuje přímou komunikaci mezi primárním a sekundárním zařízením. Velkou nevýhodou řešení je nutnost několika SPI sběrnic na řídícím modulu. Další nevýhodou je nemožnost rozšířit sběrnici o další typ sekundárního modulu.

Zkoumal jsem ještě jiná řešení komunikačních modulů a velmi mě zaujala sběrnice CAN. Sběrnice CAN se používá v běžných automobilech. Jedná se o vysokorychlostní sběrnici, která umožňuje komunikaci více zařízení. Pro implementaci jsem se rozhodl použít modul *CAN-BUS Shield V2 - high-performance MCP2515 controller and MCP2551 transceiver* [8]. Tento modul je připojený za pomocí SPI sběrnice. Na tomto modulu pak MCP2551[36] realizuje rozhraní CAN sběrnice. Na rozhraní je možno komunikovat rychlostí až 1MB/s. Tato implementace umožňuje komunikaci jak běžným (11-bit), tak rozšířeným (29-bit) protokolem. Bity v hlavičce určují prioritu zpráv na sběrnici. Každý modul má masku a filtr. Maska určuje bity v hlavičce, podle kterých se rozhoduje, zda zpráva bude přijata. Filtr určuje hodnoty těchto bitů pro přijetí zprávy.

Pro realizaci této sběrnice existují i jiné moduly. Příkladem je modul *I2C CAN-BUS Module based on MCP2551 and MCP2515* [37] Tento modul se připojí pomocí I2C rozhraní řídícího modulu. Hlavní nevýhodou tohoto modulu je menší kompaktnost. Nicméně používá I2C rozhraní, které umožňuje snadné použití modulů, které neimplementují rozhraní modulu Arduino UNO.

Pro své řešení jsem se rozhodl použít modul *CAN-BUS Shield V2 - high-performance MCP2515 controller and MCP2551 transceiver*[8]. Hlavní výhodou byla kompaktnost.



Obrázek 2.10: *CAN-BUS Shield V2 - high-performance MCP2515 controller and MCP2551 transceiver* [8]

2.3.3.2 Primární modul

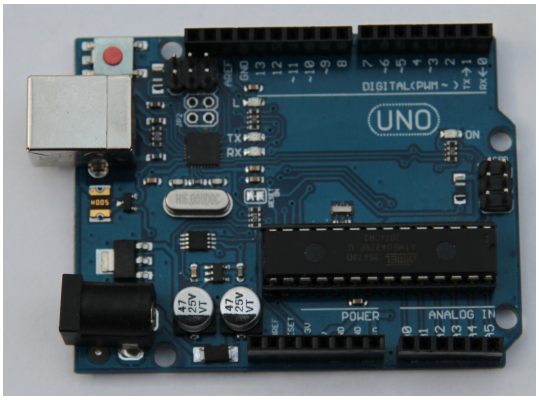
Primární modul zpracovává zprávy ze sekundárních modulů. Musí být schopen vyhodnotit data ze vstupních senzorů a poté musí být schopen pomocí zvolených komunikačních rozhraní komunikovat s uživatelem. Pro komunikaci s uživatelem využívám ve své implementaci dvou rozhraní. Prvním rozhraním je dálkové ovládání, které umožňuje pouze přijímání povelů. Druhým rozhraním je sériová linka, které umožňuje příjem povelů i hlášení chyb. Primární modul vyhodnocuje stav těchto rozhraní a udržuje vozítko v bezpečném stavu v případě jejich absence.

V řešení, která jsem nalezl, je použit pro primární modul ST Nucleo F401RE [18]. Tento modul je napájený 7-12[V]. Pracovní napětí modulu 3.3[V]. Modul využívá mikrokontroler STM32F401RET6 [19]. Kvůli pracovnímu napětí není tento modul kompatibilní s modulem, který jsem zvolil pro realizaci sběrnice CAN.

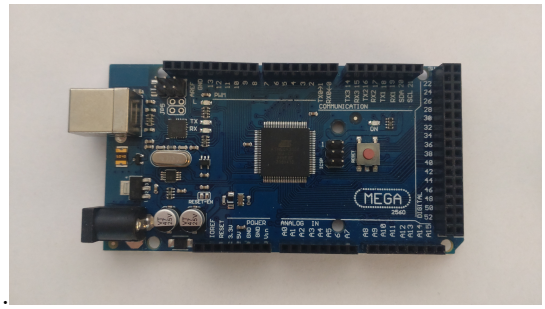
Další možností je modul Arduino UNO.[9] Ten je napájený 7-12[V]. Pracovní napětí modulu je 5[V]. Modul je postaven na mikrokontroleru ATmega-328P[38].

Další možností je modul Arduino DUE.[39] Tento modul je napájený opět 7-12[V] Jako pracovní napětí využívá 3.3[V]. Modul je postaven na mikrokontroleru Atmel SAM3X8E ARM Cortex-M3 CPU[40]. Kvůli pracovnímu napětí tento modul není kompatibilní s modulem, který jsem zvolil pro realizaci sběrnice CAN. Tento modul sám o sobě implementuje Harwarové rozhraní pro sběrnici CAN.

Další možností je modul Arduino MEGA 2560[10]. Modul je napájený 7-12[V]. Pracovním napětím modulu je 5[V]. Modul je postaven na mikrokontroleru ATmega2560 [28].



Obrázek 2.11: Arduino UNO [9]



Obrázek 2.12: Arduino MEGA 2560[10]

Pro svoji práci jsem na konec vyzkoušel, jak modul Arduino UNO[9] tak modul Arduino MEGA 2560[9]. Důvody, který modul a proč jsem zvolil, se zabývám v kapitole realizace.

2.3.3.3 Sekundární moduly

Sekundární modul se stará o chod svých periferií. Na základě povelů z primárního modulu upravuje jejich chování. Musí zaručit jejich bezpečný chod a následně poslat informace o jejich chodu primárnímu modulu. Jsou-li na tento modul zapojeny senzory, musí sekundární modul informovat primární modul o jejich čteních.

Hlavním typem takovéhoto modulu je modul řídící motory. Na každý z těchto modulů je zapojena dvojice motorů. Dvojice motorů je ovládána za pomocí H-můstku, každý z těchto modulů přímo ovládá jeden H-můstek.

Tento modul také potřebuje zpracovat zpětnou vazbu z motorů. Obecně jsou potřeba pro každý z těchto modulů dva digitální vstupy. Aspoň na jednom ze vstupů je třeba zpracovat náběžnou hranu signálu. Proto aspoň jeden z těchto vstupů musí mít možnost navázat přerušení.

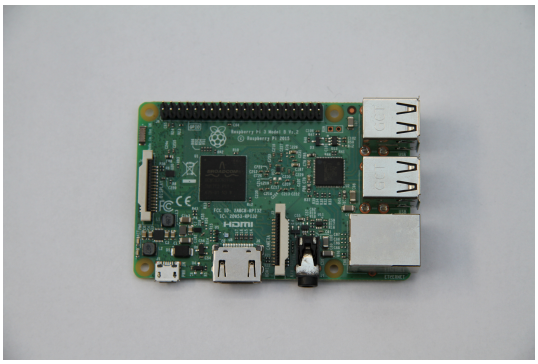
Pro výběr tohoto modulu jsou identické možnosti jako pro primární modul. Arduino UNO[9] neobsahuje dostatečný počet přerušení, aby bylo možné současně vyhodnotit signály z enkodérů a současně využít SPI ke komunikaci přes sběrnici CAN[16]. Z tohoto důvodu jsem se rozhodl zvolit pro svoji implementaci Arduino MEGA 2560[10].

2.3.4 Nadřazený modul

Nadřazený modul má být modul připojený k řídícímu modulu. K tomuto účelu je třeba využít rozhraní, které oba moduly implementují. Za pomocí rozhraní tento modul ovládá primární modul. Tento modul může být rozšířen o komplikovanější senzory a jiná komunikační rozhraní.

Pro realizaci modulu jsem se rozhodl pro platformu Raspberry Pi [17]. Platforma Raspberry Pi obsahuje mikrokontrolery a malé počítače.

Konkrétně jsem se rozhodl využít modul Raspberry Pi 3 Model B. Tento modul obsahuje Quad Core 1.2GHz Broadcom BCM2837 64bit CPU a má 1GB RAM. Implementuje několik komunikačních rozhraní. Nejdůležitější pro náš projekt budou 4xUSB 2.0, které použijeme pro komunikaci s primárním modulem. Na tomto modulu jsou implementovány rozhraní BCM43438 wireless LAN a low energy bluetooth. Tato rozhraní jsou použitelná pro rozšiřující aplikaci. Modul je napájen 5 [V] a vyžaduje proud 2.5 [A] [11].



Obrázek 2.13: Raspberry Pi 3 Model B[11]

2.3.5 Napájení

K zajištění mobility robota je třeba použít akumulátor. Akumulátor musí být schopen napájet veškeré řídící moduly umístěné na podvozku. Z tohoto akumulátoru je třeba také napájet motory.

Na původním projektu je použita dvojice Li-Pol akumulátorů propojených paralelně. Tyto akumulátory poskytovaly napětí 7,4 [V] a měly kapacitu 5200 [mAh]. Napájení v původním projektu bylo jednotné a obsahovalo pouze jeden obvod. [41]

Pro napájení mého řešení jsem použil olověný akumulátor *MB 7.2-12*. Jedná se o uzavřený gelový akumulátor. Akumulátor má napětí 12 [V] a kapacitu 7.2 [Ah]. Jeden akumulátor tohoto typu proto stačí na pohon celého podvozku. Nevýhoda je váha a velikost baterie. Proto pro ni bylo třeba vymyslet nový způsob upevnění. [42]

Obrázek 2.14: Akumulátor

Zvolené moduly pro realizaci řídícího systému je možno napájet až 12 [V] a měly by být schopny vydržet až do 20 [V], proto je lze napájet přímo z této 12[V] baterie. H-můstky mohou ovládat obvod s napětím až 35 [V].

Některé komponenty jsou dále napájeny 5 [V]. Například nadřazený modul, enkodéry nebo HY-SRF05[14]. Pro jejich napájení bylo třeba vytvořit 5 [V] obvod. Tento obvod bude napájen ze stejného zdroje. K napájení jsem použil 5 [V] výstup z platformy Arduino.

Dále je třeba měřit stav akumulátoru. Zvolené moduly pro realizaci primárního a sekundárních modulů umožňují vyčtení 0-5[V] za pomocí AD převodníku. Pro měření akumulátoru bylo třeba převést 0-12 [V] na 0-5 [V].

2.3.5.1 Raspberry Pi

Raspberry Pi model 3 B[11], které plánuji použít k realizaci nadřazeného modulu vyžaduje napájení 5 [V] a proud 2.1 [A]. Za těchto parametrů již nestačí napájení poskytované primárním modulem. K tomuto účelu je možné použít spínaný DC/DC měnič z 12 na 5 [V]. Kvůli dostupnosti jsem k účelu jsem se rozhodl použít M-Style 2x USB-A zásuvka - nabíječka 36W QC 3.0 [43]. Jedná se o USB zásuvku připojitelnou na přímo na zvolený akumulátor, která poskytuje modulu dostatečné napájení.

2.3.6 Senzory

Na vozítku je možné měřit několik různých parametrů. Nás zajímá především rychlost motorů, stav akumulátoru a vzdálenost vozítka od překážky. Rychlost

jednotlivých motorů bude měřena za použití enkodérů. K měření stavu akumulátoru je použita odporová dělička. Vzdálenost od překážky bude měřena za pomocí vzdálenostního senzoru.

2.3.6.1 Měření rychlosti motorů

U každého motoru je třeba měřit rychlost a směr otáčení. K tomuto účelu jsem se rozhodl použít enkodéry připevněné k motoru. Enkodéry jsou realizovány za pomocí magnetu a Hallova senzoru. Otáčka motoru je rozdělena na 400 jednotek.

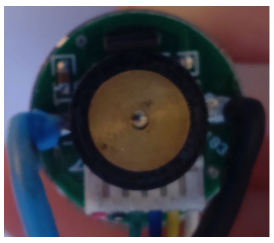
Na každém motoru je dvojice těchto senzorů. Toto umožňuje vyčíst směr otáčení motoru. Tato dvojice je od sebe fázově posunutá. Je-li druhý senzor sepnutý při náběžné hraně prvního senzoru, pak se motor otáčí po směru hodinových ručiček, v opačném případě se otáčí obráceně.

Chceme-li zjistit rychlost otáčení motoru, pak je třeba změřit, kolik náběžných hran vygeneruje Hallův senzor za časový úsek. Výsledkem je rychlost v jednotkách za časový úsek. Pro svoji aplikaci jsem se rozhodl měřit v úsecích $10[\mu s]$.

Jedna jednotka odpovídá úseku následujícího rozměru.

$$s = 2\pi r/400$$

Kde r je poloměr kola. Výsledná jednotka rychlosti s závisí na jednotce r. Na modelu r rovno 60[mm]. Výsledek tohoto přepočtu pak používám k realizaci zpětné vazby z motorů.

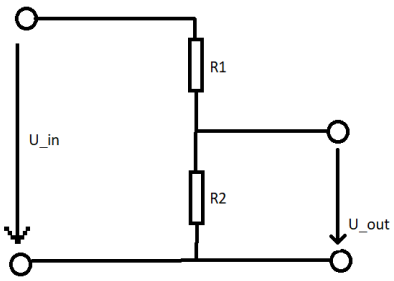


Obrázek 2.15: Enkodér

2.3.6.2 Měření stavu akumulátoru

Další měřenou veličinou je stav akumulátoru. U dobíjecích akumulátorů dochází k poklesu výstupního napětí při jeho vybíjení. Cílem tohoto měření je zabránit poškození akumulátoru jeho úplným vybitím. Všechny řídící jednotky, uvažované pro tento projekt, umožňují čtení napětí na vstupu. Toto napětí je třeba ovšem převést na rozhodovací hodnotu, kterou je možné zpracovat přímo mikrokontrolerem. Za tímto účelem jsem se rozhodl využít od-

porovou děličku. Tento obvod se skládá ze dvou odporů. Při správné volbě umožňuje vytvořit zdroj napětí o požadované hodnotě 5 [V] viz. obr. 2.16.



Obrázek 2.16: Odporová dělička

Napájecí akumulátor pracuje s napětím 12 [V]. Rozhodovací napětí většiny mikrokontrolerů je buď 5 [V] nebo 3.3 [V]. Maximální napětí, které by akumulátor v tomto systému mohl mít, je až 20 [V]. Kvůli této vlastnosti bude vhodné zvolit odpory tak, aby obvod realizoval převod z rozsahu 0 - 20 [V] na rozsah 0 - 5 [V]. Výstupní napětí obvodu lze vypočítat za pomocí následujícího vzorce.

$$U_{out} = U_{in} * R_2/(R_1 + R_2)$$

Pro realizaci zmíněného poměru je třeba, aby poměr modifikující vstupní napětí U_{in} roven jedné čtvrtině. Tohoto lze dosáhnout například při volbě odporů $R_1 = 1[k\Omega]$ a $R_2 = 3[k\Omega]$.

Na Arduinu je vyčtená hodnota mezi nulou a tisíc dvacet tři. Proto je pak třeba získat ze vstupu hodnotu za pomocí jednoduchého přepočtu.

$$U_{in} = read * (U_{max}/max)$$

Kde U_{in} je vstupní napětí, $read$ je načtená hodnota v mikrokontroleru, U_{max} je maximální vstupní napětí odporové děličky a max je maximální hodnota, kterou může mikrokontroler vyčíst.

2.3.6.3 Měření vzdálenosti

Další veličinu, kterou je třeba měřit, je vzdálenost od překážky. Tato veličina umožňuje detekci překážky. K tomu jsem použil senzor HY-SRF05[14]. Tento senzor umožňuje měřit vzdálenost s přesností na 2 [cm] a to až do vzdálenosti 450 [cm]. Informace umožňuje upravit rychlost vozítka tak, aby bylo schopno v čas zastavit před překážkou, což umožní ochranu vozítka před poškození nárazem.

Senzor vrací délku prodlevy mezi výstupním a vstupním signálem. Známým faktorem je rychlost zvuku ve vzduchu. Signál vyslaný senzorem se vrátí zpět k vozítku s časovým posunem. Pro výpočet vzdálenosti od překážky je tedy třeba vzít naměřený čas a rychlost zvuku a vynásobit jí časem. Tento výsledek je pak třeba vydělit dvěma, protože zvukový signál koná cestu k překážce a zpět.

Rychlost zvuku ve vzduchu je 343 [m/s]. Čas, za který se mi signál vrátí z modul je v [μs]. Pro získání vzdálenosti je třeba vzít tuto rychlost v [$cm/\mu s$], tedy 0.0343 [$cm/\mu s$]. Jak jsem již zmínil, na konec je třeba tuto vzdálenost vydělit dvěma. Pro výpočet je tedy použit tento vzorec, kde s označuje vzdálenost od překážky v [cm] a t je čas, naměřený modulem v [μs]:

$$s = t * 0.0343/2$$

Toto měření je třeba pravidelně opakovat, aby primární modul měl aktuální informace. Výrobce doporučuje měřit maximálně s periodou 60 [μs]. Primární modul tedy opakuje měření s touto periodou na obou senzorech.

2.3.7 Vysílač a přijímač

Pro ovládání vozítka uživatelem je použit RC ovladač. K dispozici jsem měl pětikanálový ovladač *DX5e* od společnosti Spektrum[22].

K tomuto ovladači je možné použít více různých přijímačů. Doporučovaným přijímačem je přijímač *AR600* od stejné společnosti. Jedná se o šestikanálový přijímač, takže jeden kanál na něm zůstane nevyužit. [44]

Ve zkoumaných projektech byl ke stejnému účelu použit přijímač *MK610* od společnosti Mikron. Tento přijímač zastane stejnou funkci. Přijímač je šestikanálový, tudíž jeden kanál zůstane nevyužit. [29]

Pro svůj projekt jsem již měl k dispozici přijímač *AR400* od společnosti Spektrum. Jedná se o starší model. Tento přijímač viz. obr. 2.14. podporuje pouze 4 kanály vysílačky, takže v tomto případě zůstane na ovladači nevyužit jeden kanál. [12]

Obrázek 2.17: Spektrum AR400[12]

Kromě počtu kanálů jsou všechny tyto přijímače zaměnitelné a mají i obdobný výstup.

2.3.8 Upevnění

Vzhledem ke změně systému bylo třeba vymyslet nový systém upevnění komponent na vozítku. Pro realizaci nového upevnění jsem se rozhodl využít 3D tisk.

Pro 3D tisk je třeba provést několik kroků. V první fázi je nutné vytvořit 3D model. Ten je třeba exportovat do kódu použitelného tiskárnou a následně objekt vytisknout.

2.4 Softwarové nástroje

Pro programovatelné moduly je třeba zvolit vhodná vývojová prostředí, ve kterých bude možné napsat jejich program. Upevnění modulů bude realizováno použitím 3D tisku, pro který je potřeba navrhnou modely a následně je předat 3D tiskárně. Pro tyto účely je třeba zvolit vhodné softwarové nástroje.

2.4.1 Programovací prostředí

Každý programovatelný modul bylo třeba naprogramovat. V závislosti na platformě modulu bylo třeba správně zvolit programovací prostředí. Tato prostředí v kombinaci s moduly umožňují snížení komplexity kódu využitím již implementovaných nástaveb pro specifické platformy.

2.4.1.1 Arduino

Jak primární, tak sekundární moduly, jsou založeny na platformě Arduino. Tato platforma je založena na procesorech od firmy Atmel. Firma byla odkoupena firmou Microchip, která pokračuje ve výrobě řad starých produktů firmy Atmel a rozšířila své produktové řady o tyto procesory.

Všechny procesory jsou podporovány v Microchip Studiu. Jedná se o nadstavbu na vývojové prostředí od firmy Eclipse[45]. Programování v Microchip Studiu je možné, ale vyžaduje modifikaci software. Programátor má v tomto prostředí plnou kontrolu nad mikrokontrolerem. [46]

Druhou variantou je Arduino IDE, vyvinuté pro platformu Arduino. Tato platforma umožňuje snadné psaní kódu pro Arduina jako taková. Dále umožňuje snadné čtení výpisu z takovýchto produktů. Existuje několik verzí. [47]

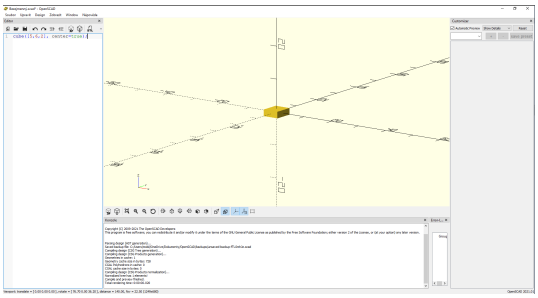
Arduino IDE umožňuje snadné nahrání a otestování kódu do procesoru. Používá zjednodušené rozhraní, které využívá knihoven, přidaných projektem Arduino. Jedná o velmi používané rozhraní, tudíž jsou pro něj snadno dostupné knihovny jiných projektů. Od verze 2.0.0 umožňuje používání nápověd v kódu, což byla před tím největší výhoda Microchip Studia.

Arduino Web Editor umožňuje psaní kódu online. Největší nevýhodou této varianty je nutnost být neustále online. Jinak je tato varianta obdobná s Arduino IDE 2.0.0.

Všechny zmíněné varianty používají jazyky C a C++, v nichž je možné provést implementaci kódu. Pro implementaci svého projektu jsem se rozhodl použít Arduino IDE. Ušetří mi práci oproti Microchip Studiu s nahráním a překladem a sníží složitost programovaného kódu.

2.4.2 3D modelování

Pro vytvoření 3D modelů tisků je možno použít program OpenSCAD viz. obr. 2.18. V tomto programu se používají předdefinované 3D objekty, které představují základní 3D tvary. Obsahuje několik základních operací s těmito objekty. Uživatel pak píše kód, který popisuje žádaný objekt. V tomto programu je výhodou, že uživatel může snadno vytvářet jednodušší objekty. Tyto objekty se poté velmi snadno a přesně skládají. [13]



Obrázek 2.18: OpenScad [13]

Alternativ k tomuto programu je mnoho. Jedním z populárních programů je Blender. Tento program je vhodnější na vytváření uměleckých objektů. Přesná manipulace s parametry modelu je v něm obtížná. [48]

Dále je možné použít množství placených programů. Tyto programy používají většinou různé typy přístupu. Dle mých znalostí se jedná o mix mezi čistě grafickým a čistě kódovým přístupem.

Pro svůj projekt jsem použil program OpenSCAD.

2.4.3 Tisknutí modelu

Model je třeba převést na kód proveditelný tiskárnou. Nejčastěji se používá formát gcode. Jedná se jednoduše čitelné instrukce pro tiskárny. Instrukce pak nastavují pozici tisknoucí hlavy, teploty zmíněné hlavy a podložky, množství vytlačeného filamentu a rychlosti větráku. K tomuto lze použít několik různých programů. [49]

Nejzákladnější z těchto programů je program Slic3r, který je opensource program vyvinutý pro Rep-Rap 3D tiskárny. Uživatel potřebuje experimentál-

ně zjistit za pomocí několika tisků správné parametry pro tisk na jeho tiskárně. [50]

Velmi dobrá, ohledně nastavení parametrů, je nástavba Průša Slic3r od firmy Průša. Nástavba má již vestavěné profily pro tisk na jejich tiskárnách. Drobnými úpravami těchto profilů se uživatel snadno dostane k parametrům tiskárny, kterou skutečně používá. [51]

Jinou alternativou je Ultimaker Cura. Tento program používá jiné rozhraní k dosažení podobných výsledků. Výhodou tohoto řešení je přidání variací v programu, které umožňují variaci v textuře vytisknutého objektu. [52]

Pro svoji aplikaci jsem používal 3D Slic3r.

2.5 Návrh

V této části projektu bylo třeba provést vlastní návrh. Návrh určuje rozvržení jednotlivých komponent v rámci systému. Určuje také, jaké funkce budou muset tyto komponenty plnit pro funkci celku.

Řídící systém se skládá ze tří typů programovatelných modulů. První je v roli uživatel, což je nadřazený modul. Primární modul sbírá informace a řídí za pomocí povelů celý systém. Závěrem je zde trojice sekundárních modulů.

Každý sekundární modul je přímo propojen s primárním modulem. K tomuto účelu navrhuji použít sběrnici CAN. Primární modul je třeba připojit k nadřazenému modulu. K tomuto účelu navrhuji použít sériovou linku. Na primární modul jsou dále zapojeny moduly sloužící jako senzory a dálkové ovládání.

2.5.1 Primární modul

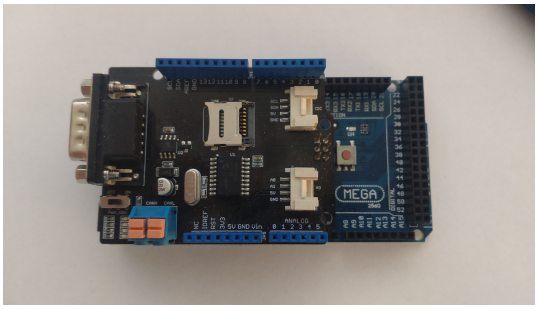
Tento modul, viz. obr. 2.21. blokové schéma, je založen na platformě Arduino 2560[10]. K modulu je zapojeno několik vedlejších modulů, které přímo ovládá.

Primární modul je zapojen na rozhraní CAN[16]. Po tomto rozhraní přijímá informace ze senzorů na sekundárních modulech. K sobě má připojené senzory a moduly k příjmu povelů od uživatele. Toto rozhraní je realizováno za pomocí modulu *CAN-BUS Shield V2 - high-performance MCP2515 controller and MCP2551 transceiver*[8] viz. obr. 2.19.

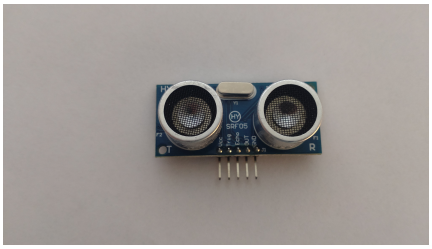
Dalšími moduly zapojenými na primární modul jsou moduly pro vnímání okolí vozítka. Tyto moduly jsou realizovány za pomocí senzoru HY-SRF05[14] viz. obr. 2.20. Primárním účelem těchto modulů je měřit vzdálenost vozítka od překážek v určených směrech, především před a za vozítkem.

Primární modul musí dále přímo zpracovávat stav akumulátoru. Pro realizaci měření stavu akumulátoru je použita odporová dělička. Dojde-li pak k přílišnému poklesu napětí na vstupu, zastaví primární modul činnost vozítka, aby nedošlo k poškození baterie.

Na primární modul zapojen přijímač AR400[12], který slouží k předání povelů od uživatele vozítka. Příjem těchto povelů je pak řešen za pomocí



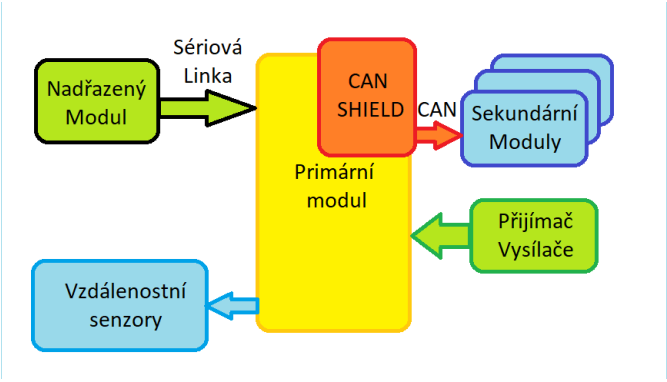
Obrázek 2.19: Arduino Mega 2560 s CAN shieldem [10] [8]



Obrázek 2.20: HY-SRF05 [14]

měření délek pulzů. V případě odpojení vysílačky modul zaznamená absenci pulzů a vyšle signál k zastavení vozítka.

Posledním modulem zapojeným na primární modul je nadřazený modul. S primárním modulem komunikuje pomocí Sériové linky. Tento modul je také druhým rozhraním, odkud mohou přicházet uživatelské příkazy. Tomuto modulu jsou předány všechny informace o stavu vozítka.



Obrázek 2.21: Struktura primárního modulu

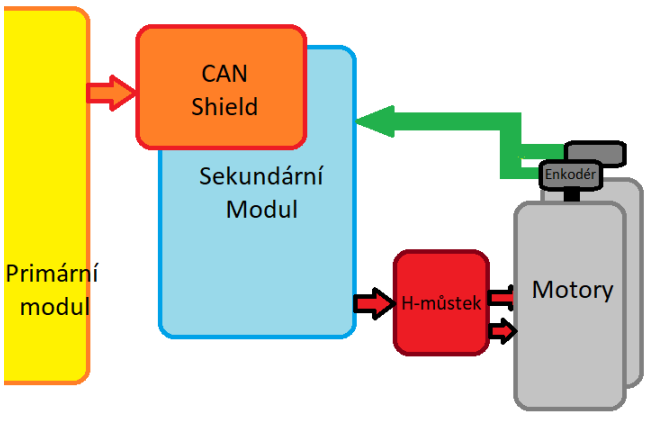
2.5.2 Sekundární moduly

Sekundárních modulů, viz. obr. 2.22. blokové schéma, je na vozítku několik. Původně jsem chtěl tyto moduly realizovat za pomocí platformy Arduino UNO[9]. Při zapojování jsem zjistil, že modul Arduino UNO nepodporuje pro všechny vyžadované funkce dostatečné množství externích přerušení. Proto jsem se rozhodl pro tento modul využít Arduino 2560[10].

Na sekundární moduly zapojené H-můstky s motory. H-můstky jako takové, slouží k ovládání motorů vozítka. Zkoušel jsem použít H-můstky L298N [6]. Výhodou tohoto můstku byla implementace 5[V] zdroje, který snadno umožňoval napájení senzorů a dalších 5[V] periferií. Tento můstek umožňuje ovládat dvojici motorů, avšak neumožňuje ovládanému obvodu vyvinout dostatečný výkon a také neobsahuje ochranu proti přehřívání. Z tohoto důvodu jsem pro ovládání vozítka zvolil jiný můstek, a to H-můstek VNHSP30 [7]. Tento H-můstek řešil oba předchozí problémy. Oba H-můstky, L298N a VNHSP30 využívají stejného způsobu ovládání. Tento H-můstek ale navíc realizuje diagnostický pin, který umožňuje další kanál zpětné vazby ovládaného motoru.

Dále jsou na sekundární modul zapojeny výstupy enkodérů z něj ovládaných motorů. Tyto enkodéry umožňují sledovat rychlost a směr otáčení motorů. Na základě těchto vstupů modul upravuje své výstupy.

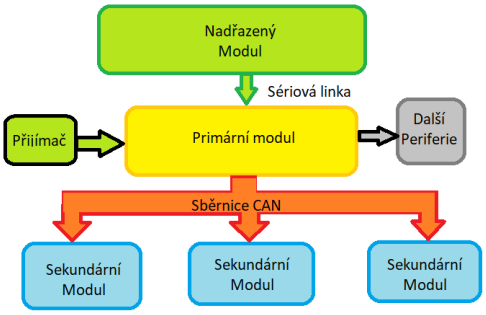
Posledním modulem zapojeným na sekundární modul je rozhraní pro komunikaci přes CAN. Tento modul je, stejně jako u primárního modulu, realizován pomocí modulu *CAN-BUS Shield V2 - high-performance MCP2515 controller and MCP2551 transceiver*[8]. Modul je k sekundárnímu modulu i k primárnímu modulu připojen za použití rozhraní Arduino UNO.



Obrázek 2.22: Struktura sekundárního modulu

2.5.3 Nadřazený modul

Nadřazený modul, viz. obr. 2.23., generuje příkazy pro primární modul. K primárnímu modulu je připojen pomocí sériové komunikace. Pro svůj projekt jsem se rozhodl využít modul Raspberry Pi 3B [11]. Jedná se o malý počítač velikosti kreditní karty. Primární modul je k tomuto modulu připojen pomocí rozhraní USB. Toto umožňuje snadno navázat sériovou komunikaci mezi oběma platformami.



Obrázek 2.23: Propojení primárního modulu

2.5.4 Komunikační protokol

Pro můj projekt bylo třeba realizovat dvě komunikační rozhraní. Prvním rozhraním je rozhraní v rámci řídícího modulu. Jak jsem se zmiňoval, primární modul musí komunikovat se sekundárními moduly. Pro realizaci tohoto úkolu jsem se rozhodl využít sběrnici CAN[16]. Na této sběrnici je třeba navrhnout obsah a priority zpráv. Dále je třeba detekovat výpadek komunikace a v takovém případě se zachovat bezpečně.

Druhým užitým rozhraním je rozhraní mezi primárním (řídícím) modulem a nadřazeným modulem. Po tomto rozhraní je třeba předávat informace o stavu jednotlivých částí vozítka. Jedná se o rozhraní, které by mělo umožnit snadné napojení na vozítko a umožnit jeho ovládání. Opět je třeba navrhnout obsah zpráv.

2.5.4.1 CAN

Pro realizaci sběrnice CAN je třeba zvolit několik parametrů. V první řadě je třeba zvolit komunikační rychlost. Dále je třeba zvolit délku hlaviček zpráv. Výhodou sběrnice, oproti běžné SPI sběrnici je možnost komunikace s více zařízeními přes jedno rozhraní a detekce chybných zpráv.

Implementace sběrnice CAN, kterou jsem zvolil, umožňuje komunikační rychlost mezi 5 a 1000 [KB/s]. Ve vozítku potřebuji rychlou odezvu, ale nechci přetížit žádný z modulů, proto jsem se rozhodl zvolit rychlost 500 [KB/s].

Je třeba vybrat mezi běžným a rozšířeným typem zpráv. Hlavní rozdíl mezi těmito dvěma typy zpráv je délka hlavičky. Délka hlavičky běžné zprávy je 11 bitů. Rozšířená zpráva má hlavičku 29 bitů. Vzhledem k tomu, že očekávám malé množství různých typů zpráv, rozhodl jsem se zvolit 11 bitovou hlavičku.

Následně bylo třeba zvolit typy zpráv, které budou chodit po této sběrnici. Mezi zprávami je třeba určit jejich prioritu, což se promítne do obsahu jejich hlaviček.

Prvním typem nejčastěji posílané zprávy bude příkaz k nastavení rychlosti motorů. Tato zpráva bude generována v primárním modulu a bude přijímána sekundárními moduly. Doručení této zprávy bude kritické. Je nutné, aby dorazila sekundárním modulům, protože se může jednat i o povel k zastavení vozítka. Obsahem této zprávy je cílená rychlost motorů. U této zprávy bude praktické, aby ji bylo možno zaslat jak jednotlivým sekundárním modulům, tak i všem modulům najednou. Tuto skutečnost je třeba zohlednit v návrhu hlavičky zpráv.

Druhým typem zprávy pak bude stav sekundárního modulu. Zde bude všechny zprávy přijímat primární modul. Je vhodné již na úrovni implementace CAN vyfiltrovat zprávy tak, aby se nimi sekundární moduly nezabývaly. Obsahem těchto zpráv by již měl být pouze zpracovaný vygenerovaný stav motoru. Tento stav by měl primárnímu modulu usnadnit práci s touto informací. Dále je třeba u této zprávy identifikovat odesílatele. Toto je možné provést buď v hlavičce nebo v datech zprávy.

Další vhodnou zprávou je zpráva pro ohlášení kritické chyby, ať už z primárního modulu do sekundárního, či ze sekundárního do primárního. U tohoto typu zprávy může být vhodné přímo informovat sekundární moduly, aby nedošlo k poškození vozítka.

Pro odlehčení užívání sběrnice je možné využít zprávu typu live. Tato zpráva by dala najevo ostatním modulům, že moduly jsou v systému a že vše je v pořádku. Užité zprávy jsou velmi krátké, tudíž je možné je použít na místo této zprávy. Další možné využití zprávy live je detekce nových modulů na sběrnici. Pro realizaci této práce jsem usoudil, že by to nebylo praktické.

Tabulka 2.1: Příkazy pro sekundární modul

Typ zprávy	kód zprávy	směr	cílová adresa	data
stop	0b0000000	0	0bXXXX (0b0000 pro všesměr)	no data
chyba	0b0000001	0	0b0000	no data
command	0b0000100	0	0bXXXX (0b0000 pro všesměr)	rychlosti

Tabulka 2.2: Zprávy sekundárního modulu

Typ zprávy	kód zprávy	směr	cílová adresa	data
stop	0b0000000	1	0bXXXX	chybový status
chyba	0b0000001	1	0bXXXX	chybový status
command	0b0000100	1	0bXXXX	rychlosti

V tabulkách 2.1 a 2.2 jsou popsány struktury zpráv na sběrnici. Pro usnadnění jsem použil 0bXXXX pro označení adres jednotlivých sekundárních zařízení, kterých se zpráva týká. Rychlosti se posílají dvě a každá rychlost zabírá dva Byty. Chyba je jedno Bytové číslo. Je-li chyba 0, vše je v pořádku. Je-li chyba 1, přišel příkaz k uvedení do chybového stavu. Je-li jiná chyba, pak chyba nastala přímo na modulu sekundárním modulu s danou adresou. Zpráva ze sekundárního zařízení je vždy podepsaná odesílatelem. Navrhovaný obsah zpráv podporuje maximálně 15 sekundárních modulů.

2.5.4.2 Sériová komunikace

Sériová linka bude sloužit ke komunikaci mezi primárním a nadřazeným modulem. Je třeba po ní přenést informace dvěma směry. Jednak povely k pohybu určeným směrem pro primární modul. Primární modul by měl být schopen z této žádosti vypočítat rychlosti motorů, které zašle sekundárním modulům. Na druhou stranu by měl primární modul podat na vyžádání od nadřazeného modulu zprávu o stavu vozítka a jeho senzorů.

Pro použití této komunikace jsem se rozhodl využít rozhraní, která jsou již implementovaná jak na primárním, tak na nadřazeném modulu. Arduino MEGA 2560[10] implementuje USB B rozhraní[53]. Toto rozhraní se dá velmi snadno propojit s USB portem na Raspberry Pi[17].

Na obou zařízeních je pak implementovaný UART protokol[32]. Tento protokol umožňuje komunikaci pomocí osmibitových zpráv. Každá taková zpráva obsahuje jeden kontrolní bit. U tohoto protokolu je dále třeba zvolit baudrate. Pro svoji aplikaci jsem zvolil 115200 [Bd].

Pro přehlednost komunikace je vhodné celou zprávu skládat z více menších zpráv. Je to vhodné ze dvou hlavních důvodů. Prvním je velikost zprávy. Pro ovládání vozítka, ale i obdržení informace o stavu vozítka, je třeba více než osm bitů. Dále je vhodné kontrolovat korektnost doručených informací alespoň kontrolním součtem.

V návrhu protokolu jsem proto zvolil startovní symbol, který značí začátek zprávy. Dále jsem identifikoval typ zprávy a vyčetl z ní data. Na konec ještě kontroluji validitu odeslané zprávy jako celku. K tomu používám kontrolní součet.

Prvním typem zprávy je nastavení směru pohybu vozítka. Zde jsem zvolil formát zprávy obdobný výstupu z vysílačky, to znamená dvou dimenzionální bod se souřadnicemi v rozmezí od -100 do +100. Primární modul na tuto zprávu pak pošle odpověď s těmito hodnotami, aby bylo možno vyhodnotit korektnost přijaté zprávy a případně ji poslat znova.

Dále je zde žádost o informaci o stavu vozítka. Nadřazený modul tuto zprávu obdrží a odešle zpět nadřazenému modulu stav vozítka. Součástí této zprávy jsou i informace o vzdálenostech od překážky. Zpráva neobsahuje již stavy jednotlivých motorů, ale jen stav vozítka jako celku. Senzory HY-SRF05 mohou měřit vzdálenost do 450 [cm] s přesností na 0.3 [cm], proto bude vhodné tuto informaci správně zakódovat.

V rámci tohoto protokolu je vhodné používat zprávu bez dat k ověření přítomnosti nadřazeného zařízení. V případě, že primární modul neobdrží reakci od nadřazeného modulu, musí vyhodnotit jeho nepřítomnost. V takovém případě se musí adekvátně zachovat a případně zastavit vozítko. Zároveň je vhodné na takovou zprávu poslat odpověď. Neobdrží-li primární modul odpověď po dlouhý čas, měl by uživateli nahlásit chybu.

2.5.5 Program

Každý ze zmíněných modulů je řešen programovatelným mikrokontrolerem. Jedná se o systém propojený po sběrnicích. Každý z modulů řídícího systému potřebuje mít definované chování.

Prvním část chování, kterou potřebuje mít definovanou, je bezpečný stav. Toto je stav, do kterého se modul dostane při spuštění systému či detekované poruše v jiné části systému.

Druhým stavem je běžné chování. V tomto stavu bude modul řídit své periferie dle definovaných parametrů a bude kontrolovat, zda v řízeném systému nedošlo k chybě. V případě nastalé chyby přeskočí modul do bezpečného stavu, obstará všechny své periferie a poté nahlásí chybu nadřazenému systému.

V této sekci budu pokračovat, tak jak jsem prakticky návrh prováděl. V první části jsem vytvořil podřazené moduly. Poté na základě informací, které pro řízení těchto modulů byly relevantní, jsem vytvořil nadřazené moduly.

2.5.5.1 Sekundární modul

Sekundární moduly přímo ovládají motory. Mají také přímo přístup k výstupům z motorů. Na každý z těchto modulů je zapojen modul pro ovládání motorů VNH3SP30[7]. Každý z těchto modulů umožňuje ovládání dvou motorů. K

modulu je zapojena zpětná vazba z motorů. Pro tento konkrétní systém se jedná o dvojici výstupů z enkodérů.

První částí tohoto programu bude zpracování zpětné vazby z enkodérů. Enkodér je nastavený tak, že při točení spíná a rozpíná svůj výstup v závislosti na otáčení motorů. Počet těchto sepnutí je přímo převeditelný na úhel otočení rotoru od začátku měření. Druhý výstup pak spíná stejným způsobem, ale je o půl fáze vychýlený. Proto, sepnou-li výstupy enkodéru současně, jedná se otočku jedním směrem, nesepnou-li jedná se o otočku druhým směrem. Výstup lze použít ke korekci směru otáčení rotoru. Poté je možné za pomocí časovače uvnitř mikrokontroleru spočítat rychlost otáčení motoru. Pro získání rychlosti jsem zvolil časový úsek 10 [μs]. Z tohoto jsem vyhodnotil rychlost v otáčkách za sekundu, kterou používám jako vstup řídící soustavy. K načítání rychlosti jsem se rozhodl využít vnějšího přerušení, které vždy vyvolám na náběžnou hranu přicházející ze senzoru.

Řízení každého motoru obsahuje tři výstupy. Prvním je dvojice výstupů určující směr otáčení motorů. Tyto výstupy se chovají následovně. Jsou-li oba rozepnuty, motor je zastaven, ale dá se s ním volně hýbat. Jsou-li oba sepnuty, motor je zastaven, ale při pokusu o otáčení vnější silou klade odpor. Je-li sepnut jen jeden z těchto výstupů, pak se motor točí rychlostí určenou třetím vstupem. Pro realizaci těchto výstupů používám digitální výstup.

Dalším výstupem potřebným pro ovládání výstup je PWM signál. Tento signál určuje procento výkonu motoru. PWM, neboli pulzně šířková modulace, je periodická modulace založená na dvou hodnotách. Modulace pak nabývá vždy v periodě střídavě jedné a druhé hodnoty. Procentuálně se pak určuje, v jaké hodnotě a po jakou část periody signál bude. V řídícím modulu je tento signál řízen hodnotou mezi 0 a 255, kde 0 znamená trvale rozepnuto a 255 trvale sepnuto.

Pomocí těchto tří signálů se ovládá obvod VNH3SP30[7]. Na tento obvod je zapojen obvod s motory, který je regulován právě pomocí těchto tří signálů.

Zpětnovazebný systém je celý realizovaný v programu. Pro každý motor je zde realizovaný samostatný systém. Regulovanou hodnotou systému je trojice signálů, to jest dvojice signálů pro nastavení směru otáčení a PWM signál. Signály pro ovládání směru mohou nabývat pouze hodnot 0 a 1. PWM signál může nabývat celočíselných hodnot mezi 0 a 255. Výstupní hodnotou tohoto systému je rychlost v otáčkách za sekundu. Očekávaná hodnota, která do systému přichází, je také v této jednotce. Program provede žádané změny na řízených hodnotách a snaží se přiblížit žádané hodnotě rychlosti.

PWM přímo ovládá výkon motoru, proto je možné omezit jeho výkon v případě prokluzu. Je-li naopak motor nadměrně zatížen, systém se pokusí jeho směrem poslat více výkonu, aby bylo dosaženo žádané rychlosti.

Požadovaná hodnota vychází ze stavu, ve kterém se systém nachází. Za bezpečný stav jsem se rozhodl prohlásit stav, kdy jsou motory zastaveny. Tudíž žádaná rychlost v tomto stavu je 0 otáček za sekundu.

Za běžného stavu cílená hodnota vychází z povelu obdrženém po sběrnici CAN[16]. Systém se snaží co nejvíce přiblížit žádané hodnotě. I při běžném provozu může přijít příkaz k okamžitému zastavení, který má vyšší prioritu, než ostatní rychlosti.

V případě, že výstup enkodérů moc dlouho neodpovídá, musí sekundární modul vyhodnotit poruchu systému. Jedná se speciálně o stav, kdy je na motor daný výkon nějaký čas a není detekován žádný pohyb. V tomto případě musí sekundární modul uvést motory do bezpečného stavu a o této události zpravit primární modul.

Pro detekci chyby a stavu obsahuje modul VNH3SP30[7] obsahuje piny DIAG. Je-li na těchto pinech detekována chyba, systém se opět musí uvést do bezpečného stavu a zpravit o chybě nadřazený modul.

Poslední detekovatelnou chybou je výpadek komunikace s nadřazeným modulem. Výpadek je detekován prostřednictvím časovače. Nedojde-li včas žádná zpráva od primárního zařízení, je vyhodnocen výpadek komunikace a systém je uveden do bezpečného stavu.

2.5.5.2 Primární modul

Primární modul řídí komunikaci v rámci řídícího modulu. V první řadě zpracovává příkazy od uživatele. K tomuto se používají dva způsoby ovládání, RC přijímač a sériová linka.

Jedním způsobem ovládání je ovládání za pomoci vysílačky DX5e Spektrum [22]. Výstupy z tohoto kanálu se přijímají ve formě výstupu z AR400 [12]. Vzhledem k tomu, že se jedná o ovladač primárně určený pro modely letadel, obsahuje tento ovladač více kanálů, než je pro řízení vozítka potřeba. Pro řízení vozítka jsem použil dva kanály. Z těchto kanálů pak primární modul vypočítává dvojici rychlostí. Každá z těchto rychlostí je určena pro jednu stranu vozítka. Tyto rychlosti pak primární modul zašle sekundárním modulům.

Výstupem modulu AR400 je PWM signál, který je třeba převést na čas. V případě odpojení ovladače pak tento signál skončí. Toto je třeba detekovat.

Druhým způsobem ovládání je komunikace pomocí sériové linky s nadřazeným modulem. Zde přijme primární modul příkazy v podobné formě jako z vysílačky. V tomto případě primární modul musí informovat o stavu vozítka nadřazený modul. Toto provádí stejným kanálem. Není-li přítomen žádný ze způsobů řízení, musí primární modul udržovat vozítko v bezpečném stavu, tedy ho zastavit.

Na primární modul jsou dále připojeny senzory HY-SRF05[14]. Pomocí těchto senzorů primární modul zjišťuje vzdálenosti od překážek a vyhodnocuje přijaté příkazy. V případě možnosti kolize modifikuje jejich parametry.

Sám o sobě tento modul provádí měření stavu akumulátoru. K tomuto je na analogový vstup tohoto modulu zapojena odporová dělička. Touto cestou pak primární modul dostává informace o výstupu akumulátoru. Napájení je

touto cestou převedeno z napětí akumulátoru na rozhodovací napětí mikrokontroleru, tedy 0 až 12 [V] na 0 až 5 [V]. Kvůli odolnosti systému jsem zvolil maximální napětí až 20 [V], ale neočekávám, že takto vysoké napětí se vyskytne v systému. Klesne-li napětí pod stanovenou úroveň modul vyšle příkaz k zastavení chodu motorů a informuje všechny moduly o této události.

Pomocí sběrnice CAN jsou k primárnímu modulu zapojeny sekundární moduly. Primární modul těmto modulům pravidelně posílá požadovanou rychlost a monitoruje jejich odezvu. Dojde-li k absenci odezvy déle než 30 [ms], je třeba uvést systém do bezpečného stavu. V případě chybové zprávy musí řídící modul udělat totéž.

Kdykoliv uvádí řídící modul ostatní moduly do bezpečného stavu, je třeba o této události dát vědět nadřazenému modulu. Nadřazený modul také musí být informován o zdroji této chyby, aby byl uživatel schopen se zachovat adekvátně.

Primární modul, jako celek, musí převést informace z jednotlivých sekundárních modulů na sjednocený stav celého systému. Toto slouží k vyhodnocení stavu vozítka jako celku. Dále tento stav slouží k hlášení nadřazenému modulu. Není žádoucí, aby byl nadřazený modul informován o stavu každé části řízeného systému. Proto jsou mu hlášeny běžně pouze stavy celku a chyby pouze na vyžádání.

2.5.5.3 Nadřazený Modul

Nadřazený modul se chová z hlediska primárního modulu jako jeden ze dvou uživatelů. Komunikaci mezi primárním a nadřazeným modulem jsem se rozhodl řešit formou request-respond. Z toho plyne, že nadřazený modul si musí u primárního modulu zažádat o informace, nebo zadat nový směr pohybu pomocí sériového protokolu. Poté musí počkat na odpověď od primárního modulu. Na základě těchto odpovědí pak vyhodnotí nový žádaný směr.

Bude-li detekována chyba primárním modulem, tato informace se k nadřazenému modulu dostane ve formě odpovědi na zprávu. Program v tomto modulu bude udržovat zprávu o této chybě a měl by si o ní vytvořit záznam. Modul tuto zprávu pouze uloží k dalšímu zkoumání.

Kapitola 3

Realizace

Navržený systém bylo třeba realizovat. V této kapitole procházím postup realizace řídícího systému. Při realizaci projektu bylo třeba nejprve upravit podvozek a poté koupit, upevnit, zapojit a naprogramovat vybrané moduly.

3.1 Zapojení

Každý modul potřebuje specifické napájení. Celé vozítko je napájené z baterie, proto pro každý modul bylo třeba realizovat napájení. Dále bylo třeba propojit komunikační rozhraní těchto modulů..

3.1.1 Zdroj napájení

Jako hlavní zdroj energie na vozítku jsem se rozhodl využít olověný akumulátor *megaBat R-AKU.12V-7AH* [42]. Jedná se o stejnosměrný zdroj s napětím 12 [V]. Pro většinu řídících modulů lze tento zdroj použít bez jakékoliv modifikace.

Operační napětí použitých modulů je však mezi 0 a 5 [V]. Moduly pro snímání okolí HY-SRF05[14] vyžadují napájení 5 [V]. Modul AR400[12] je možné napájet od 3.5 do 9.6 [V], takže není možné přímé napájení z akumulátoru. Z těchto důvodů bylo třeba do projektu přidat 5 [V] zdroj. Původně jsem používal H-můstkek L289N [6], který obsahuje zdroj 5 [V]. Tyto moduly jsem používal i k napájení řídících modulů. Vzhledem k absenci tepelné ochrany a maximální zátěži 2[A] na kanál se tento modul ukázal pro projekt nevhodný. Proto bylo třeba najít použitelný zdroj. Dalším modulem, který obsahuje zdroj 5 [V] je modul Arduino Mega 2560 Rev. 3[10], který jsem se rozhodl použít pro realizaci primárního a sekundárních modulů. Tento modul je napájen přímo z akumulátoru 12 [V].

K napájení Raspberry Pi [11], který jsem se rozhodl použít pro nadřazený modul, je třeba napětí 5[V] a proud 2.1 [A]. Toto už je moc velká zátěž na primární modul. Z tohoto důvodu je nadřazený modul napájen samostatně

za použití M-Style 2x USB-A zásuvka - nabíječka 36W QC 3.0 [43], která je zapojena přímo na akumulátor.

Vznikly tedy tři okruhy. Prvním okruhem je 12 [V] obvod napojen přímo na akumulátor. Obvod je monitorován za pomocí odporové děličky, která umožňuje odečíst případný nežádoucí pokles napětí. Na tento obvod jsou přímo zapojeny všechny řídící moduly Arduino Mega 2560 Rev. 3 [10] a jsou z něj napájeny všechny motory za použití H-můstku VNH3SP30[7]. H-můstek tento obvod reguluje. Jako zdroj 5[V] pro méně náročné je periferie je použit primární modul. Nadřazený modul je napájen 5[V] samostatným okruhem, jehož zdroj je M-Style 2x USB-A zásuvka - nabíječka 36W QC 3.0 [43], který je napájen přímo z baterie.

3.1.2 Komunikační sběrnice

Pomocí sběrnic je nutno zajistit propojení jednotlivých modulů. Některé ze sběrnic je možno zapojit více způsoby. Pro každý modul bylo třeba zvolit rozhraní a následně vybrat formu jeho propojení.

3.1.2.1 CAN

Sběrnice CAN v rámci řídícího modulu propojuje primární a sekundární moduly. Tato sběrnice je realizována v systému moduly *CAN-BUS Shield V2 - high-performance MCP2515 controller & MCP2551 transceiver* [8]. Modul je následně v systému propojen pomocí rozhraní Arduino UNO[9], které je realizované i na Arduinu MEGA 2560[10]. Propojení je realizováno použitím SPI[20] rozhraní na tomto modulu viz. tabulka 3.1.

Moduly jsou pak mezi sebou propojeny za použití dvou kabelů. Jeden z těchto kabelů je signál LOW a druhý signál HIGH. Pro propojení těchto kabelů z tohoto modulu je možné použít dvě rozhraní, buď pomocí konektoru DB9, nebo použitím svorkového terminálu.

Prvním rozhraním realizovaným na modulu emphCAN-BUS Shield V2 - high-performance MCP2515 controller & MCP2551 transceiver [8] je CAN_L a CAN_H. Z tohoto terminálu jsou pouze dva vývody. Rozhraní propojuje pouze tyto dva signály.

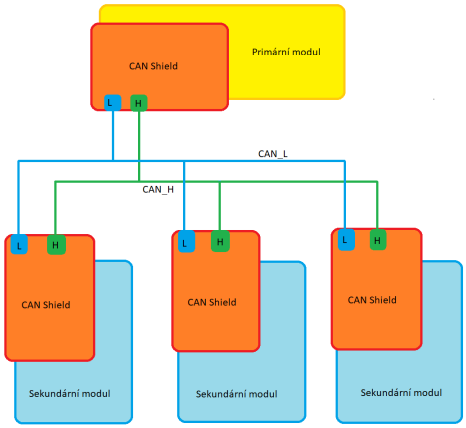
Dalším rozhraním použitelném je konektor DB9. Toto rozhraní umožňuje opět propojení za pomocí dvou výše zmíněných signálů CAN_L a CAN_H. Rozhraní umožňuje napájení modulu za pomocí pinů V_OBD a GND. Pro svůj projekt jsem se rozhodl využít toto rozhraní. Moduly jsou již napájeny z akumulátoru. Proto jsem použil pouze piny CAN_L a CAN_H viz. obr. 3.1.

3.1.2.2 Seriová komunikace

Sériová komunikace je použita mezi primárním a nadřazeným modulem. Primárním modulem je modul Arduino[10]. Nadřazeným modulem je modul Raspberry Pi[11]. Na každém z těchto modulů existují dva typy rozhraní,

Tabulka 3.1: Komunikační rozhraní pro CAN Shield

Funkce	Arduino MEGA 2560 pin
SCK	D13
MISO	D12
MOSI	D11
CS	D9



Obrázek 3.1: CAN síť

které je možné využít k realizaci sériové linky. Sériová linka se skládá ze dvou signálů. Jedná se signály RXD a TXD.

Na Arduinu je možno použít USB-B konektor či pouze rozhraní pinů. Rozhraní USB-B umožňuje napájení Arduina bez dalších modifikací. USB-B také umožňuje přímé přeprogramování primárního modulu při dodržení protokolu.

Rozhraní přes piny se skládá pouze ze dvou signálů. Tyto signály jsou přístupné na běžném rozhraní Arduina UNO.

Raspberry Pi obsahuje dvě možnosti, jak se k němu připojit sériovou linkou. První možností je rozhraní pomocí USB portu. Raspberry Pi obsahuje čtveřici portů USB-A, jejichž prostřednictvím je možno tuto komunikaci realizovat.

Dále je na Raspberry Pi hřebínek s přímo vyvedenými piny. Mezi těmito piny jsou i piny pro realizaci sériové linky.

Rozhodovací úroveň Raspberry Pi je 3.3[V], kdežto u užitého Arduina je 5[V]. Pro využití rozhraní realizované hřebínky na obou stranách, je třeba přidat 3.3V/5V level-shifter, který ochrání Raspberry Pi. Pro jednoduchost jsem se rozhodl použít USB rozhraní na obou modulech.

Tabulka 3.2: Ovládací piny VNH3SP30[7]

Funkce	Arduino MEGA 2560 pin (sekundární modul)
PWMA	D5
DIRA1	D8
DIRA2	D7
PWMB	D6
DIRB1	D4
DIRB2	D10
DIAGA	A2
DIAGB	A3

Tabulka 3.3: Zapojení měření baterie

Funkce	Arduino MEGA 2560 pin (primární zařízení)
V_{in}	A0

3.1.3 Motory

Každý z motorů ovládá jedno kolo a je regulován za použití H-můstku VNH3-SP30[7]. Každý H-můstek reguluje dva motory a je řízen sekundárním modulem. Tento modul je zapojen pomocí Arduino UNO rozhraní. Zapojení řídících signálu z Arduina je popsáno v tabulce 3.2. Obvod regulovaný H-můstkem je napájený přímo z akumulátoru. Řídící logika H-můstku je napájena ze sekundárního modulu přes rozhraní Arduino UNO.

3.1.4 Senzory

Každý ze senzorů potřebuje zpracování svých výstupů a napájení. Výstupy jsou zpracovávány příslušným modulem.

3.1.4.1 Měření stavu akumulátoru

Měřenou veličinou je napětí, měření provádí primární modul. Použitá platforma Arduino používá rozhodovací napětí 5[V]. Arduino, za pomocí AD převodníku, je schopno měřit napětí, nicméně pouze do své rozhodovací úrovně. Měřený obvod operuje na úrovni 12 [V]. Bylo třeba vytvořit obvod, který by napětí převedl na rozhodovací úroveň mikrokontroleru. Za tímto účelem jsem použil odporovou děličku[54]. Jedná se o obvod, který je schopen rozdělit napětí dle poměru odporů. Pro bezpečný provoz jsem se rozhodl použít děličku, která zmenšuje napětí až z 20[V] na žádanou úroveň 5 [V]. Obvod je realizován za pomocí dvou odporů. Tento obvod je zapojen paralelně k ostatním zařízením přímo na akumulátor. Výstup je zapojen na řídící modul viz. tabulka 3.3.

Tabulka 3.4: Zapojení enkodérů

Funkce	Arduino MEGA 2560 pin (sekundární modul)
ENCAA (interrupt)	D18
ENCAB	D19
ENCBA (interrupt)	D20
ENCBB	D21

3.1.4.2 Enkodéry

Enkodéry jsou součástí modulu motoru. Enkodéry pro každý motor jsou realizovány za pomocí dvojic halových čidel a magnetu, který je připevněn přímo k rotoru motoru. Tyto enkodéry vyžadují napájení 5 [V]. Enkodér každého motoru má dva výstupní signály, které jsou zapojeny na vstupy sekundárního modulu viz. tabulka 3.4.

Pro měření rychlosti otáčení motoru je nutno detekovat náběžnou hranu aspoň na jednom výstupu z těchto senzorů. Pro detekci směru otáčení je následně třeba zkontrolovat stav druhého senzoru na náběžné hraně. Je-li tento signál sepnutý, jedná se o jeden směr, v případě rozepnutého signálu, jedná se o směr opačný.

3.1.4.3 Vzdálenost

Pro detekci vzdálenosti jsem se rozhodl použít senzor HY-SRF05 [14]. Jedná se ultrazvukový modul. Modul je řízen signálem TRIGG. Pro spuštění měření je třeba tento signál sepnout po dobu 10 [μs]. Tím se spustí měřící sekvence senzoru. V průběhu měření senzor vygeneruje osm zvukových vln. Poté čeká, kdy se vrátí jejich ozvěna. Senzor změří prodlevu mezi odeslaným a přijatým signálem. Po provedení měření senzor odešle výsledek pomocí signálu ECHO do primárního modulu. Na signálu ECHO je vytvořena vlna. Délka vlny odpovídá vzdálenosti od překážky.

Pro projekt je použita dvojice těchto modulů. Každý z modulů sleduje jeden směr pohybu. Senzor dává primárnímu modulu zprávu o vzdálenosti od překážky na své straně. Zapojení na primární modulu je popsáno v tabulce 3.5.

Tabulka 3.5: Zapojení vzdálenostních senzorů

Funkce	Arduino MEGA 2560 pin (primární modul)
TRIGGF1	D14
ECHOF1	D15
TRIGGF2	D48
ECHOF2	D49
TRIGGF3	D46
ECHOF3	D47
TRIGGB1	D44
ECHOB1	D45
TRIGGB2	D42
ECHOB2	D43
TRIGGB3	D40
ECHOb3	D41

Tabulka 3.6: Zapojení modulu AR400[12]

Funkce	Arduino MEGA 2560 pin (primární modul)
RUDD	D3
ELEV	D4
AILE	D5
THRO	D6

3.1.4.4 Příjem z ovladače

Pro projekt jsem použil kombinaci ovladače DX5e[22] a přijímače AR400[12] od firmy Spektrum. Dálkový ovladač DX5e[22] je běžný modelářský ovladač, primárně určený k ovládání letadýlek na dálkové ovládání. Tento ovladač má celkově čtyři výstupy. Ovladač má oddělené napájení od zbytku vozítka, realizované pomocí čtveřice AA baterií 1.5 [V].

Do systému je přímo zapojen modul AR400[12], který má 4 výstupy. Pro ovládání vozítka jsou zapotřebí dva. Tyto výstupy jsou zapojeny na primární modul viz. tabulka 3.6

Je-li k tomuto modulu připojen dálkový ovladač, modul vysílá opakovaně vlny na své výstupy, které přímo odpovídají polohám páček na dálkovém ovladači. V případě, kdy je ovladač odpojen, modul přestane tyto vlny vysílat. Vzhledem k tomu, že je třeba měřit šířku vlny, je zapotřebí, obdobně jako u senzoru HY-SRF05[14], modul zapojit k digitálním pinům primárního modulu.

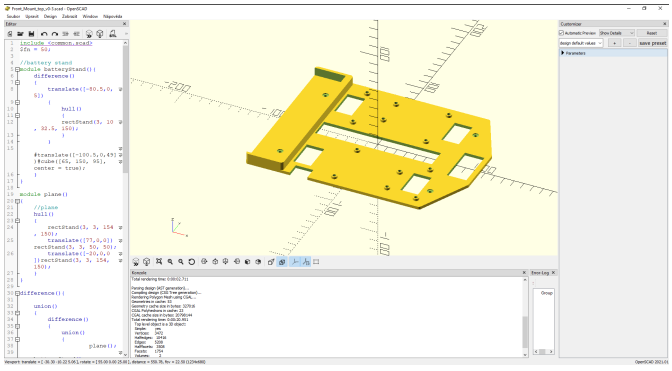
Tento modul není možné napájet 12[V] akumulátorem. Doporučené napětí je mezi 3.5 a 9.6 [V]. Proto, obdobně jako u dalších senzorů, je modul napájen napětím 5[V].

3.2 Upevnění

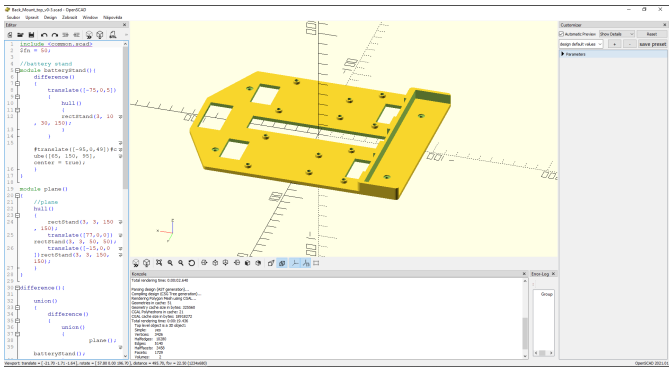
Na podvozek bylo třeba připevnit všechny části. Největší problém byl s akumulátorem. Akumulátor je těžký a velmi ovlivňuje rozložení váhy vozítka. Proto bylo třeba akumulátor umístit do středu. Následně bylo třeba vyřešit umístění ostatních modulů.

Pro umístění akumulátoru jsem vytvořil speciální platformu, kterou jsem připevnil k podvozku. Na tuto platformu jsem umístil i primární a sekundární moduly. Pro účely 3D tisku byla navržená platforma příliš velká, proto jsem ji rozdělil na dvě části viz. obr. 3.2. a 3.3. .

Pro návrh platformy jsem použil program Openscad[13]. V tomto programu jsem si vytvořil moduly pro realizaci jednotlivých součástek. Následně jsem je složil v celek.



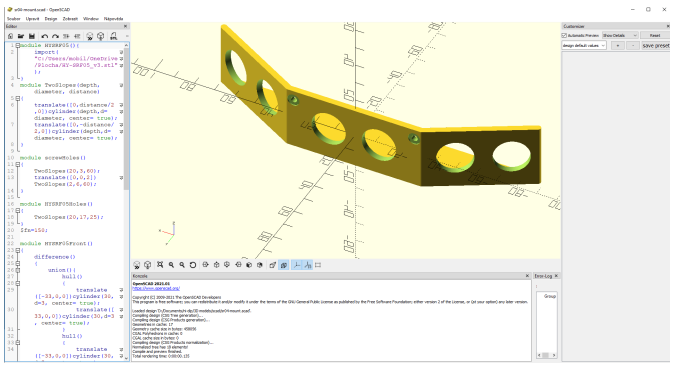
Obrázek 3.2: Upevnění modulů na přední části vozítka



Obrázek 3.3: Upevnění modulů na zadní části vozítka

Dále bylo třeba vyřešit uchycení senzorů HY-SRF05[14], které snímají kužel před vozítkem. Tento kužel má úhel patnáct stupňů. Uchycení je navrženo tak, aby se tyto úhly doplňovaly. Modul uprostřed snímá překážky přímo před

vozítkem. Moduly po stranách jsou vytočeny o třicet stupňů, což umožňuje snímání překážek po stranách viz. obr. 3.4.



Obrázek 3.4: Upevnění modulů HY-SRF05[14]

3.3 Program

Každý z programovatelných modulů musel mít svůj vlastní program. Pro moduly bylo třeba zvolit správný programovací jazyk a prostředí. V závislosti na vstupech a výstupech má každý z modulů definované chování.

3.3.1 Programovací prostředí

K programování modulů založených na platformě Arduino jsem použil programovací prostředí Arduino IDE[47]. Jedná se o zjednodušené prostředí. V tomto prostředí se používají jazyky C a C++. Prostředí umožňuje snadný přístup ke knihovnám. Na začátku projektu byla k dispozici pouze verze 1.8.0. Tato verze obstarala pouze základní funkcionalitu. V průběhu realizace projektu byla vydána verze 2.0.0, která umožnila debugování kódu a nápovědy při psaní kódu.

3.3.2 Řízení motorů

Motory jsou řízeny sekundárními moduly. Každý ze sekundárních modulů ovládá dvojici motorů, které jsou propojeny s řídícím modulem za použití H-můstku. Výkon motoru je ovládaný za pomocí PWM signálu. Dále pomocí dvojice signálu INA a INB je určen směr otáčení motoru.

Z VNH3SP30[7] je vyveden diagnostický výstup. Tento výstup je v logické jedničce, pokud nedošlo k poruše. V případě, že k ní dojde, změní se hodnota tohoto výstupu na logickou nulu. Tento výstup umí detekovat dva typy poruch. Jednou možnou poruchou je přehřátí modulu. Druhou možnou chybou je

detekce zkratu na diagonále H-můstku. V případě, že k jedné z těchto poruch dojde, je třeba motory zastavit a informovat nadřazený modul.

3.3.3 Regulace motoru

Pro regulaci otáček je třeba realizovat zpětnovazebný obvod. Vstupem tohoto obvodu je rychlost a směr otáčení motoru. Každý motor je za tímto účelem vybaven enkodérem, který je realizovaný dvojicí Hallových senzorů. Z výstupů těchto senzorů je počítán směr otáčení. K vyčtení rychlosti se počítá počet signálů za určený časový úsek.

Změřená rychlost a směr otáčení jsou vstupní hodnotou regulátoru. Druhou vstupní hodnotou regulátoru je žádaná rychlost motoru. Zde se jedná o proměnou nastavenou pomocí příkazů z ostatních částí programů.

Regulátor reguluje proměnné PWM, INA a INB, viz. řízení motoru, pro každý motor. Směr je regulován za pomocí znaménka rychlosti. Neshodují-li se znaménka, pak je třeba otočit směr otáčení.

Výkon motoru reguluje PWM. Rychlost otáčení motoru je závislá na zátěži motoru. Systém se snaží dosáhnout cílené rychlosti. Je-li rychlost dále od 0, než současná rychlost regulátor zvýší hodnotu PWM. Je-li tomu obráceně, regulátor sníží hodnotu PWM.

Při regulaci je možné za pomocí konstanty měnit přesnost regulátoru. PWM nabývá hodnot 0 až 255. Nejmenším možným krokem regulace je 1/255 výkonu. Tento krok je možné zvětšit pro méně přesnou, ale rychlejší regulaci.

Pro samotnou regulaci jsem se rozhodl měnit přesnost regulace v závislosti na rozdílu cílené a současné rychlosti. Je-li tento rozdíl velký, provádím skoky o velikosti 10/255. Blíží-li se rychlost chtěné rychlosti, změním velikost kroku na přesnější hodnotu 1/255.

3.3.4 CAN

Komunikace mezi primárním a sekundárními moduly je realizovaná za pomocí sběrnice CAN[16]. Zvolené řešení se skládá ze dvou modulů MCP2551[36] a MCP2515[55]. Modul MCP2551[36] realizuje fyzikální vrstvu dle standartu ISO-11898. Výstupem tohoto modulu jsou dva signály a to RXD a TXD. Signál RXD převádí reprezentuje rozdíl mezi CAN_L a CAN_H. Signál TXD detekuje příchozí zprávu na sběrnici. Alternativně je tento signál využit pro detekci chyby na sběrnici.

Modul MCP2515[55] realizuje CAN V2.0B s rychlostí 1 [Mb/s]. Modul obsahuje dva přijímací buffery a umožňuje příjem zpráv o délce až 8 bytů. K přijetí zpráv se používá kombinace masky a filtru. Maska nastavuje bity v hlavičce, které je třeba pro přijetí zprávy porovnat. Filtr určuje hodnoty, kterých mají tyto bity nabývat. Dojde-li ke schodě bitů hlavičky zprávy s maskou a filtrem, zpráva je přijata do bufferu. Modul je ovládán za použití SPI rozhraní.

Moduly MCP2551[36] a MCP2515[55] jsou součástí modulu *CAN-BUS Shield V2 - high-performance MCP2515 controller and MCP2551 transceiver* [8]. Modul umožňuje komunikaci s mikrokontrolerem pomocí SPI rozhraní. Pro Arduino byla k modulu dodána knihovna. Tato knihovna implementuje komunikaci potřebnou k použití modulu.

Pro použití rozhraní je třeba nastavit několik věcí. V první řadě je třeba zvolit přerušení pro přijetí zprávy. Přerušení je možné vyvolat na dvou pinech. Při dodání je přerušení na pinu 9 z pohledu Arduino UNO rozhraní. Po provedení HW změny na modulu je možné toto přerušení přepojit na pin 10. V případě, že je na tomto pinu náběžná hrana, znamená to, že modul přijal validní zprávu a tato zpráva je připravena v bufferu ke zpracování.

Na zvoleném rozhraní je třeba nastavit rychlost na sběrnici. Knihovna implementuje komunikaci několika rychlostmi. Maximální rychlostí je 1 [Mb/s]. Knihovna umožňuje použití specifických rychlostí mezi 5 [kb/s] a 1 [Mb/s]. Pro realizaci projektu jsem se rozhodl použít rychlost 500 [Kb/s]. S touto rychlostí je třeba rozhraní inicializovat.

Dále je nutné nastavení masek a filtrů. Zpráva se skládá z několika částí. Nejdůležitější části jsou datová část a hlavička. Hlavička určuje prioritu zpráv na sběrnici. Všechny moduly se pokusí začít komunikaci současně. Modul odesílající zprávu s nejnižší hodnotou hlavičky na sběrnici zprávu odešle. Všechny moduly pak současně zprávu přijímají. Dojde-li ke shodě hlavičky s permutací masky a filtrů, zpráva je modulem přijata a zapíše se do bufferu ke zpracování.

Jak jsem se zmínil v části návrhu, pro svůj projekt jsem se rozhodl použít běžnou délku hlavičky, tj. 11-bit. Pro přijetí je třeba určit prioritu doručení zpráv. K určení typu zprávy jsem použil prvních šest bitů hlavičky. Čím nižší toto číslo je, tím vyšší má zpráva prioritu na sběrnici. Moduly musí zpracovat všechny zprávy určené pro něj, nezávisle na jejím typu, tudíž tyto bity jsou maskami ignorovány. Na jejich základě se pak příjemce rozhoduje, kterou zprávu přijal.

Další bit jsem se rozhodl použít pro určení směru zprávy na sběrnici. Rozhodl jsem, že zprávy od primárního modulu mají větší prioritu na sběrnici, než zprávy pro řídící modul. To umožňuje snadné nastavení přijímací masky a filtru na primárním modulu. Všechny zprávy pro řídící modul budou mít tento bit nastavený na jedničku. Také při příjmu libovolné zprávy na tomto bitu záleží. Z toho vyplývá, že ve všech přijímacích maskách musí být tento bit nastaven na logickou hodnotu jedna. Ve filtrech pak pro primární modul musí být také tato hodnota nastavena na jedničku. Ve filtrech pro sekundární moduly je naopak tento bit nastaven na nulu.

Poslední čtyři bity zprávy jsem se rozhodl použít pro identifikaci sekundárního modulu, kterému byla zpráva míněna, nebo od kterého zpráva pochází. Pro tento účel jsem každému sekundárnímu modulu na sběrnici přiřadil číslo větší nebo rovno jedné. V současnosti jsou na této sběrnici tři sekundární moduly, tudíž se jedná o binární reprezentace čísel 1,2 a 3. Pro zrychlení

komunikace jsem se také rozhodl odlišit zprávy určené všem sekundárním modulům najednou. Zpráva určená všem sekundárním modulům má adresu 0. Sekundárním modulům z tohoto požadavku vznikají dva filtry. Prvním filtrem je filtr pro individuální zprávy. Zde se tyto bity skládají z binární reprezentace přiděleného čísla. Ve druhém filtru jsou na těchto místech nuly.

Vzhledem k obsazeným vstupům zařízení, jsem se rozhodl tuto adresu nastavit ručně při vytváření kódu. Za tímto účelem je možné využít několik digitálních vstupů. Tyto vstupy musejí být sepnuty již při zapnutí modulu. Tuto adresu neplánuji měnit za běhu systému. Hlavním důvodem je změna filtru na sběrnici CAN.

Pro samotné přijetí zprávy na primárním modulu tyto bity nejsou podstatné. Nicméně jsem se rozhodl za jejich pomocí třídit zprávy dle odesílatele. Primární modul musí vytřídit, od kterého sekundárního modulu zprávu přijal.

Zpracování zpráv se dále liší v závislosti na tom, zda se jedná o primární, či sekundární modul. Obecně ale platí, že je-li zpráva přijata pomocí těchto nastavení, modul se musí zabývat jejím zpracováním.

3.3.4.1 Zpracování zpráv: Sekundární modul

Jak jsem se již zmínil dříve, sekundární moduly přijímají dva typy zpráv. Zprávy určená přímo modulu a zprávy určené všem sekundárním modulům. Z hlediska zpracování není důležité, zda zpráva byla individuální nebo určena všem modulům. Sekundární modul si kontroluje, že primární modul běží. Sekundární modul očekává od primárního modulu zprávu s prodlevou 30 [ms]. Neobdrží-li zprávu od primárního modulu, znamená to, že komunikace byla přerušena, a je třeba se uvést do bezpečného stavu. Neučinil-li by tak, hrozí poškození vozítka.

V průběhu běžné komunikace se modul většinou setká s příkazem, nastavující žádané rychlosti motorů. Modul zprávu rozpoznává dle hlavičky při zpracování zprávy. Dále musí zkontrolovat, že souhlasí délka dat s formátem zprávy pro nastavení rychlostí. Nesouhlasí-li, zprávu zahodí. Dle svého vnitřního stavu a této zprávy sekundární modul upraví své chování. Není-li ve stavu, kdy tuto zprávu může přijmout, znamená to dvě věci. V sekundárním modulu byla detekována porucha a hlášení o této poruše se nerozšířilo do primárního modulu. V takovém případě musí sekundární modul opakovat pokus o odeslání chybového hlášení. Je-li vše v pořádku, stačí předat nové žádané rychlosti oběma zpětnovazebným regulátorům a odeslat zprávu s nově přijatými rychlostmi primárnímu modulu.

Další příkaz je příkaz k akutnímu zastavení běhu motorů. Tento příkaz se používá primárně při detekci chyby v systému. Opět zde sekundární modul nastaví zpětnovazebný modul na rychlost 0. Reakce na tuto zprávu musí být okamžitá. Po obdržení této zprávy sekundární modul vygeneruje zprávu s novým stavem. Hlavní poruchy v systému, které mohou nastat je vybití

akumulátoru, poškození motoru a výpadek komunikace. Ve všech případech pokračování chodu znamená možnost poničení zařízení.

V případě nastalé poruchy musí modul vygenerovat chybové hlášení. Hlavně se jedná o typ poruchy ke které došlo na motorech. Tuto zprávu modul odešle bez příkazu primárního modulu. V první řadě se modul sám o sobě uvede do bezpečného stavu. Tuto zprávu odešle jednou. Poté, přijde-li povel k nemožné operaci, modul tuto zprávu zopakuje. Na sekundárním modulu mohou nastat dvě chyby. První je porucha na motorech, detekovaná stavem pinu DIAGx. Tento stav modul nemůže opustit, dokud je pin DIAGx rozepnutý. Z tohoto vyplývá, že modul nemůže provádět operace, dokud porucha není odstraněna. Detekovanými poruchami je zkrat na H-můstku a jeho přehřátí. Za použití enkodérů je detekováno přetížení motoru. Toto nastane, je-li rychlost moc nízká i při plném výkonu motoru po moc dlouhý čas. Přetížení motoru může vést k jeho poškození. Poslední poruchou je jiné, než očekávané znaménko u rychlosti. Tato chyba může detekovat chybné zapojení motoru, enkodérů, či jejich poruchu. V každém případě je modul opět uveden do bezpečného stavu a vygenerována zpráva.

3.3.4.2 Zpracování zpráv: Primární modul

Primární modul, jak bylo zmíněno, průběžně zpracovává hlášení stavů jednotlivých sekundárních modulů. Tyto informace musí pak uspořádat do snadno zpracovatelného stavu. V případě, že k ní dojde, musí uvést všechny sekundární moduly do bezpečného stavu. Ze zpráv primární modul identifikuje původce poruchy. Tuto informaci si uloží a informuje nadřazený modul.

Za běžného provozu modul primární modul generuje žádané rychlosti jednotlivých stran vozítka, které posílá sekundárním modulům Za tímto účelem běžně používá univerzální zprávu. Mají-li motory od uživatele speciálně nastaveny individuální rychlosti, primární modul musí informovat individuální moduly přímými zprávami. Na tyto zprávy modul očekává odpověď. Neobdrží-li odpověď, zprávu opakuje. Jedinou výjimkou je, je-li detekována v systému porucha. V tomto případě se ujistí, že všechny moduly přijaly zprávu o chybě a jsou uvedeny do bezpečného stavu.

Z výše uvedeného plyne nastavení filtrů a masek primárního modulu. Primární modul zpracovává zprávy nezávisle na jejich typech. Pro urychlení komunikace ignoruje všechny zprávy určené pro sekundární moduly. Co se adres týče, zpracovává zprávy od všech modulů. Tato rozlišení používají stejný formát, jako u dat přijímaných sekundárními moduly. Pro primární modul platí, že zpracovává všechny zprávy, které mají nastavený směrový bit na jedna.

Pří zpracování primární modul provádí několik operací. V první řadě rozliší typ doručené zprávy. Dle této informace, pak zkontroluje korektnost doručené datové zprávy. Následně rozliší informace dle zdroje. Tato informace je uložena v adresních bitech zprávy, při čemž adresa 0 znamená nevalidního odesílatele.

Pro každý sekundární modul udržuje primární modul status, který je uložen do struktury. Pro každý sekundární modul je uložena informace o rychlosti a stavu. Je-li jediná informace pouze rychlost, pak to znamená, že na sekundárním modulu nenastala chyba. Přijde-li ze sekundárního modulu chybová hláška, pak primární modul uvádí zbytek vozítka do bezpečného stavu. Za tímto účelem primární modul vytváří stav vozítka, jako celku. O této informaci informuje primární modul všechny ostatní moduly.

Primární modul dále monitoruje komunikaci se sekundárními moduly. Podobně, jako u sekundárního modulu, přerušení komunikace s modulem je vážná chyba. V případě absence odpovědi od sekundárního modulu je třeba vyhodnotit chybu na tomto modulu. Tato chyba vyžaduje přegenerování stavu vozítka a informování všech zbývajících modulů.

Pro tyto informace má primární modul vytvořenou datovou strukturu. V této struktuře se upravují informace o jednotlivých modulech dle doručených zpráv. Pro každý sekundární modul je v této struktuře uložena žádaná rychlost. Dále pro každý z těchto modulů je uložen stav. Z těchto jednotlivých stavů pak primární modul vytváří celkový stav vozítka. Tento celkový stav modifikuje příkazy, zaslané jednotlivým modulům. Nadřazený modul je informován o tomto stavu. Nadřazený modul si může v případě poruchy zažádat o informaci o jejím zdroji. Tuto informaci primární modul vyvodí ze stavů jednotlivých sekundárních modulů.

3.3.5 Měření vzdálenosti

Primární modul zajišťuje bezpečný chod vozítka. Mezi rizika provozu patří náraz do překážky. Pro zjištění překážky je třeba znát vzdálenost od překážky.

Za tímto účelem je k primárnímu modulu připojeno několik senzorů HY-SRF05[14]. Každý z těchto senzorů registruje překážky na jedné straně vozítka. Vozítko z této informace vyvozuje, jak vzdálená je překážka od vozítka. V případě, že je překážka moc blízko, je třeba zpomalit vozítko na bezpečnou rychlost.

Senzor HY-SRF05[14] je ultrazvukový měřič vzdálenosti. Tento senzor je napájen 5 [V] a umožňuje registraci překážky ve vzdálenostech mezi 2 a 450 [cm]. Měření je prováděno s přesností na 3 [mm].

Modul je ovládán jedním pinem TRIGG, který řídí kdy je prováděno měření. Pro započetí měření je třeba tento vstup nastavit na rozhodovací napětí po dobu 10 [μs]. V první části modul vygeneruje osm zvukových vln na frekvenci 40 [KHz]. Po vygenerování sady signálů modul čeká na odezvu a měří čas. Výstupním signálem tohoto modulu je signál ECHO. Na signálu ECHO po provedení měření je vygenerován signál, jehož délka odpovídá času, po kterém modul zaznamenal první odezvu. Pro vyčtení odezvy je v programu použita funkce *pulseIn()*. Tato funkce umožňuje detekci a změření délky pulsu na digitálním vstupu.

Modul musí vozítko zastavit, daným směrem, je-li překážka blíže než 3 [cm]. Jedná se vzdálenost, do které je možné pokládat čtení ze senzorů za validní. Podrobnější informace o užití modulu HY-SRF05 lze nalézt v odkazu [56].

3.3.6 Zpracování výstupů RC ovladače

Pro projekt je použito dálkové ovládání Spektrum DX5e [22] a radiový přijímač AR400 [12]. Výstup modulu AR400 je ve formě PWM signálů. U RC modelů se tyto signály používají přímo k ovládání motorů nebo servo motorů. PWM signál na každém výstupu přijímače odpovídá poloze příslušné osy páčky na dálkovém ovladači. Pro ovládání vozítka je třeba tyto výstupy zpracovat.

Modul umožňuje zpracování čtyř výstupů, které jsou označeny THR, AIL, ELE a RUD. AIL, ELE a RUD vstupy se chovají stejně. Páčka se na těchto osách vrací na střed a používají i stejné rozsahy. Výstup THR má posunutou 0, má zmenšený rozsah a nevrací se do původní polohy.

Pro ovládání jsem se rozhodl využít vstupy ELE a RUD. Tyto vstupy jsou na kolmých osách jedné páčky. Z tohoto výstupu je možno spočítat vektor žádaného směru vozítka. Dvě polohy se přepočítají v primárním modulu na vektor směru, kterým chce uživatel vozítko pohnout. Z vypočteného směru se potom spočítají rychlosti jednotlivých motorů.

V případě absence připojeného ovladače přijímač přestane vysílat signál. Toto je měřeno primárním modulem. Naměří-li vstupní modul absenci přijímače, pak je třeba vozítko okamžitě zastavit.

Na ovladači zbývají dva volné kanály. Tyto kanály jsou běžně určeny pro ovládání parametrů letadélek. Pro moji aplikaci nejsou využity. V budoucnu je možno je použít pro rozšíření ovládání.

3.3.7 Sériová linka

Pro propojení mezi primárním a nadřazeným modulem je použita sériová linka, kterou je možné použít k ovládání a získávání celkového stavu vozítka. Primární modul na tomto kanálu očekává příchozí komunikaci. Sám o sobě modul neinicializuje žádnou komunikaci.

Primární modul detekuje výpadek na této lince. Není-li přítomný ani nadřazený modul, ani AR400[12], je třeba vozítko zastavit.

Po tomto rozhraní je komunikováno několik informací. První informací, kterou může vozítko obdržet, je vektor pohybu žádaný nadřazeným modulem. Tento vektor jsou dvě hodnoty, které jsou zpracovány stejným stylem jako výstup z vysílačky. Každá z těchto hodnot představuje souřadnici bodu ve 2D prostoru. Primární modul z těchto souřadnic vypočítá žádanou rychlost vozítka. Rychlost je dále modifikována za pomocí výstupů ze senzorů pro detekci překážky. Odpovědí na zprávu je informace o stavu vozítka.

Nadřazený modul si může zažádat pouze o stav vozítka. Tento stav je vyvozen ze stavů všech sekundárních modulů. Je-li vše v pořádku, primární modul vygeneruje zprávu o rychlostech každé ze stran.

Nastala-li chyba, primární modul vrátí celkový stav vozítka. Pro stav individuálních sekundárních modulů musí Nadřazený modul vygenerovat přímý dotaz. Primární modul na tuto zprávu generuje informace ze struktury, ve které má uložené stavy sekundárních modulů. Opět odpoví stavem, ale tento stav je rozšířen o identifikaci sekundárního modulu, který danou chybu vyvolal. Pro urychlení komunikace se počítá, že tato zpráva bude použita až teprve tehdy, bude-li vozítko jako celek uvedeno do bezpečného stavu.

Druhou běžnou informací, o kterou si může nadřazený modul zažádat, je stav senzorů. Primárně se jedná o vzdálenosti od překážek. Nadřazený modul může tyto informace použít k mapování prostředí nebo úpravě svých rozhodnutí.

Primární modul musí dále detekovat absenci nadřazeného modulu. Nepřijde-li po sériové lince po daný čas zpráva, pak je vyhodnocena chyba. Výpadek tohoto modulu změní stav vozítka, pouze pokud je nepřítomen i modul AR400.

Zpráva na sériové lince je realizována za pomocí jednoduchého protokolu. Protokol obecně definuje začátek a konec zprávy. Na začátku protokolu je startovací znak. Tento znak je následován typem zprávy. Dle typu zprávy je určena i délka zprávy. Následuje datová část zprávy. Délka této části zprávy závisí na typu zprávy. Zprávy obsahující pouze stav mají pouze jeden Byte dat. Další součástí zprávy jsou rychlosti a polohy. Na každou z těchto informací jsou třeba 2 [Byty]. Tudíž zpráva s oběma informacemi je 4 [Byty] dlouhá. Zpráva je zakončena kontrolním součtem. Jedná se pouze o XOR všech Bytů zprávy. Tento Byte je použit pro kontrolu korektnosti doručené zprávy. Není-li doručena odpověď na dotaz včas, počítá se, že zpráva nebyla správně doručena. V takovém případě musí Nadřazený modul dotaz opakovat.

Komunikace musí být správně nastavena. Na sériové lince se mění baudrate a parametry komunikace. Jako baudrate jsem se rozhodl využít 115200.

Dále na každém modulu je třeba zvolit rozhraní, které bude použito k realizaci této komunikace. K realizaci na Arduinu[15] je možné použít několik sériových linek. Pro svoji implementaci jsem se rozhodl použít linku, která je napojena na programovací konektor modulu. Toto rozhraní je realizováno za pomocí USB rozhraní[53]. Konkrétně je využit USB-B konektor.

Pro programování tohoto rozhraní je použito rozhraní Serial[32]. Toto rozhraní je nejprve třeba inicializovat za pomocí funkce *Serial.begin()*. Následná komunikace se provádí za pomocí funkcí *print()* a *read()*. Funkce *print()* odešle přímou reprezentaci bufferu a funkce *read()* pak přijme tento buffer.

Na straně nadřazeného modulu je k dispozici několik možných rozhraní. Původně jsem chtěl použít rozhraní hřebínku modulu Raspberry Pi, nicméně modul pracuje s rozhodovací úrovní 3.3 [V]. Pro využití tohoto rozhraní je třeba použít level-shifter. Proto jsem se rozhodl použít jedno z implemento-

vaných USB-A rozhraní. Toto rozhraní je již připraveno na úroveň 5 [V] takže na tomto místě není třeba žádné modifikace.

K přístupu k tomuto rozhraní z Raspberry Pi je možné použít rozhraní souborového systému, které je přístupné v operačním systému Raspberry Pi. Pro programování nad tímto rozhraní jsem ve svém projektu použit C++. Tento program užívá prosté rozhraní pro čtení a zápis do souboru. Modul opakovaně generuje dotazy. Následně ukládá informace doručené pomocí tohoto rozhraní.

Kapitola 4

Testování

Testování komponent probíhalo v několika fázích. V první fázi probíhalo testování jednotlivých komponent, které jsem se rozhodl použít pro realizaci mého řešení. Postupně jsem tyto moduly skládal do větších celků, které realizují propojení menších komponent. Každý z těchto celků bylo třeba otestovat. Na konec bylo třeba otestovat vozítko jako celek.

V první řadě bylo třeba si vytvořit předpoklad testu. Tento předpoklad zachycoval, jaké parametry je třeba ověřit. Následně bylo třeba implementovat část obvodu, která test realizuje. Po provedení testu je následně třeba vyvodit závěr. Skončil-li test neúspěchem, bylo třeba pozměnit zařízení a test zopakovat.

4.1 Ovládání motorů

Ovládání motorů je podle návrhu napojeno na sekundární modul. K ovládání motoru předpokládám využití modulu, který realizuje H-můstek a ovládá motor. U motoru je pak třeba měnit nastavení tří parametrů, které jsou popsány výše. Tyto parametry jsou DIRA, DIRB a PWM. Následně je třeba pozorovat chování motoru. Při nastavení parametrů očekávám pohyb motoru, který bude odpovídat nastaveným parametrům. Následná změna těchto parametrů musí vést ke změně chování motoru. Test ověřuje chování H-můstku, výkon a spotřebu motoru.

Jako sekundární modul je použit modul, založený na platformě Arduino. Pro měnění parametrů při testu jsem využil počítače a možnosti propojení Arduina s Windows za pomocí sériové linky, jejíž prostřednictvím přímo nastavím zmíněné parametry. Sériová linka na Arduinu umožňuje snadné poslání celého čísla se znaménkem. Signál PWM může nabývat pouze celých hodnot mezi 0 a 255. Signály DIRA a DIRB umožňují pouze celé hodnoty a to buď 0 nebo 1. Signály musí být nastaveny tak, že maximálně jeden z nich je nastavený na 1. Pro nastavení těchto dvou signálů jsem využil znaménko poslaného čísla. V této části testu ještě nebylo podstatné, kterým směrem se motor točí.

Nicméně bylo třeba odpozorovat zda při prohození nastavení DIRA a DIRB signálů se změnil směr otáčení motoru.

Modul s H-můstkem bylo třeba příslušným způsobem napájet. Z napájení je napájen obvod, který můstek ovládá. Napájení by mělo odpovídat napětí, které obdrží motor z akumulátoru. K napájení jsem používal laboratorní zdroj 0-30V/0-2,5A EP-613[57].

4.1.1 Modul L298N

Původně jsem tento test dimenzoval na využití modulu L298N[6]. Tento modul je běžně dostupný H-můstek, který byl napojen na modul Arduino UNO[9]. Na Arduinu jsem zvolil pin, který je schopný generovat PWM a dva digitální výstupy k realizaci zapojení. Modul L298N jsem pak napájel stejnosměrným napětím 12 [V].

Do Arduina jsem nahrál jednoduchý program, který se choval, jak jsem popsal výše. Následně jsem začal měnit parametry. A motor měnil chování, dle očekávání.

Nicméně spuštěním motoru na plný výkon se ukázala nedostatečnost modulu L298N pro tento projekt. Modul se začal velmi zahřívat a výkon motoru poklesl. Toto pozorování prokázalo, že tento modul není vhodný k dalšímu použití v projektu.

4.1.2 VNH3SP30

Vzhledem k nevhodnosti modulu L298N bylo třeba najít alternativu. Jako alternativu jsem se rozhodl použít modul VNH3SP30 [7]. Tento modul umožňuje větší výkon na linku, než modul L298N a zároveň modul obsahuje ochranu proti zkratu a přehřátí.

U tohoto modulu je třeba dále zapojit piny pro kontrolní signály. Očekávání je, že piny budou sepnuté za provozu.

S tímto modulem už nedocházelo k přehřívání. Poté bylo možno zapojit druhý motor. Je očekáváno, že z jednoho modulu budu ovládat dva motory. S tímto modulem byl test již úspěšný.

4.2 Zpětná vazba z motorů

Pro realizaci zpětné vazby z motorů jsou využita Hallova čidla, která jsou přímo připevněná k rotorům motoru vozítka. Cílem testu je změřit míru otočení rotoru. Hallovo čidlo ve spojení s magnetem připojeným na osu rotoru realizuje enkodér. Hallovo čidlo pak bude spínat obvod periodicky v závislosti na úhlu otočení magnetu. Toto sepnutí je třeba detekovat.

Na každém motoru je připojena dvojice Hallových čidel, která jsou vzájemně fázově posunutá. To způsobí, že nastane-li náběžná hrana na jednom z čidel, je možné ze stavu druhého čidla vyčíst, kterým směrem se motor otáčí. Z

Hallových čidel je čtena relativní pozice rotoru od počáteční pozice. Při otočení o určený úhel se čidlo vždy sepne příslušný počet krát. V závislost na směru otáčení lze pak pozici přičítat, či odečítat. Vezme-li se pak změna této pozice, za pevný časový úsek, je možné vypočítat rychlost otáčení motoru. Tímto způsobem je možné změřit rychlost motoru v daném časovém okamžiku.

Pro tento test je zapotřebí napájet pouze Hallovy senzory a sekundární modul. Pro napájení sekundárního modulu při tomto testu lze využít opět vestavěné USB rozhraní. Hallovy senzory je pak vhodné napájet přímo ze sekundárního modulu napětím 5 [V].

Program tohoto modulu potřebuje pro každý z měřených motorů dva vstupy. Na jednom z těchto vstupů je třeba detekovat náběžnou hranu. Při detekci náběžné hrany je pak třeba detekovat hladinu napětí na druhém výstupu. Pro detekci náběžné hrany na Arduinu je vhodné využít přerušení. K detekci stavu pak stačí libovolný vstupní pin.

Pro zjištění výsledku testu pomocí sériové linky posílám načtený počet přímo do počítače. Očekávám, že načtené číslo bude přímo odpovídat úhlu o který jsem rotor otočil od započetí testu. Dále očekávám, že otočím-li rotor jedním směrem číslo bude růst a otočím-li ho druhým směrem číslo bude klesat.

Při provádění tohoto testu jsem zjistil, že Hallův senzor na jednom z použitých motorů byl poškozen. Tento senzor nefungoval. Tyto senzory jsou napevno přidělané k motorům. Proto bylo třeba vyměnit jeden z motorů. Z cenových důvodů jsem zvolil výše zmíněnou novou alternativu.

Po výměně motoru již test proběhl na všech motorech úspěšně. Jelikož na jednom modulu bude zapojena dvojice motorů, je třeba pak tento test provést s dvojicí motorů. Je tedy třeba vybrat druhý pin s přerušením a další, který je možné číst. Test byl pak také úspěšný.

Rychlost motoru je pak zjištěna za časový úsek. Pro provedení tohoto testu bylo třeba přidat do programu časovač, který zajistí přepočet úhlu na rychlost za časový úsek. Počítači pak pomocí sériové linky neposílám polohu, ale rovnou přepočítanou rychlost. Očekávám, že netočí-li se rotor, přečtená rychlost bude 0. Rychlost bude růst, hýbu-li s rotorem rychleji, a klesat, hýbu-li rotorem pomaleji. Test proběhl úspěšně a hodnoty odpovídaly očekávání.

4.3 Zpětnovazebný obvod

Tento test zahrnuje již velkou část funkce sekundárního modulu. Podobně, jako u předchozích testů, je vhodné začít test pouze s jedním z motorů. Pro zapojení je třeba zkombinovat zapojení obou předchozích testů.

Sekundárnímu modulu je pak třeba zasílat informaci o žádané rychlosti motoru. K tomuto opět lze využít sériovou linku. Zpětnovazebný modul se již zabývá pouze rychlostí motoru. V závislosti na naměřené rychlosti obvod upravuje upravit svoje výstupy.

Při testu zde lze sledovat několik proměnných. Za prvé při nastavení rychlosti motor by se měl na dané rychlosti ustálit. Ustálení může trvat delší než vhodnou dobu. Vozítko by mělo na tyto změny pohotově reagovat, proto je-li prodleva mezi zadáním nového povelu a jejím provedením příliš dlouhá, jedná se o neúspěch. Je-li motor pod zátěží a je-li zpomalen, modul by měl v bezpečné míře zvýšit výkon, kterého se snaží motor dosáhnout, což se projeví velkým zvýšením PWM. Naopak při odstranění zátěže by se měl motor držet co nejblíže cílené rychlosti.

V tomto obvodu je třeba, aby byl implementovaný bezpečný stav. V tomto obvodu bezpečný stav představuje zastavení motorů. Uvedení do bezpečného stavu může přijít z více zdrojů. V první řadě může přijít přes sériovou linku, která pro tento test zastupuje primární a další moduly, nebo dojde-li k rozpojení kontrolních výstupů na VNH3SP30 [7] což detekuje zkrat nebo přehřátí motoru.

Po zapojení dle předchozích parametrů bylo třeba změnit několik výstupů. Arduino UNO [9] umožňuje pouze dvě vnější přerušení na specifických pinech. Proto je třeba mít tyto piny volné. Piny 0 a 1 jsou využity pro sériovou linku, která komunikuje s počítačem, proto je není možné využít. Vzhledem k obsazení jsem pro další fáze již změnil modul na Arduino MEGA 2560[10].

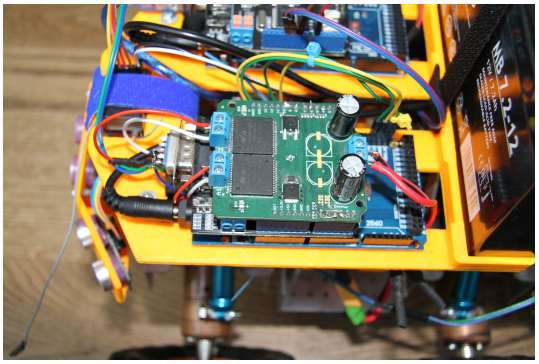
Testy proběhly dle očekávání. Pro rychlou regulaci bylo třeba zkrátit dobu měření rychlostí motorů na 10 [μs]. Z této hodnoty bylo třeba pak rychlost přepočítat na rychlost za [s]. Jedná se jednoduchý výpočet.

Dále v tomto testu jsem zjistil že používat stejně velký krok na úpravu PWM při různých hodnotách rozdílu žádané a cílené rychlosti, není vhodné. Využívání stejně velkého kroku vede buď ke ztrátě přesnosti, dostane-li se modul blíže k cílené rychlosti, nebo vede k moc pomalé odezvě na změnu rychlosti. Pro opravu tohoto rozdílu jsem se rozhodl PWM upravovat jinak, a to v závislosti na rozdílu mezi žádanou a aktuální rychlostí. Nejmenší krok, kterým můžu regulovat výkon motoru, je jedna dvě stě dvaceti pětina. Je-li absolutní rozdíl větší, než určená hodnota, bylo vhodnější používat kroky, které odpovídají deseti dvě stě dvaceti pětinám. Je-li rozdíl menší použiji menší kroky. Což vede k plynulejšímu ovládání motorů při zachování maximální možné přesnosti rychlosti.

Reakci na okamžité zastavení je možno realizovat dvěma způsoby. Jedním je nastavení cílené rychlosti a tudíž i PWM na 0. Tím přestane vozítko posílat výkon do motoru. Dále je vhodné nastavit oba ovládací piny motoru na 0. Tím se motor zastaví nezávisle na PWM. Tuto reakci jsem se v závislosti na výsledku testu rozhodl přidat, protože to umožňuje rychlejší zastavení motorů.

4.4 CAN komunikace

Pro komunikaci mezi moduly na spodní vrstvě jsem se rozhodl využít sběrnice CAN. K otestování této sběrnice je třeba propojení aspoň dvou modulů. Mezi



Obrázek 4.1: Zapojený sekundární modul

těmito moduly je třeba navázat komunikaci.

4.4.1 Testovací propojení

K zapojení této sběrnice je v celkovém návrhu použito SPI rozhraní, které je přístupné na specifických pinech Arduina. V tomto testu bylo třeba otestovat, že každý modul přijímá jemu určené zprávy.

Pro otestování je tedy třeba vytvořit malou síť. Dále jsem vypisoval na sériovou linku data přijata každým z modulů. Také bylo třeba vytvořit malou sadu testovacích zpráv, které moduly mezi sebou posílaly. Vypisoval-li každý modul, co měl na počítači, vše fungovalo správně. Výpis byl prováděn za pomocí sériové linky.

Účelem tohoto testu je vyzkoušet, zda navržené propojení modulů funguje. Následně je třeba nasimulovat zatížení linky a zjistit, zda při očekávaném zatížení bude sběrnice systému stále fungovat. Také při testu je nutné vyzkoušet očekávané množství zpráv.

Test proběhl úspěšně. Komunikace probíhala dostatečně rychle. Moduly zvládaly zprávy zpracovávat včas. Z testu nevyplynuly žádné nutné modifikace systému.

4.4.2 Propojení s sekundárními moduly

Po předchozím testu bylo třeba otestovat systém, ve kterém se již nacházely implementace testovacích modulů. Pro tento test bylo třeba vytvořit zjednodušený primární modul a připojit jej na počítač pomocí sériové linky. Tato jednoduchá implementace pouze kopíruje doručené zprávy z počítače. Z počítače posílám primárnímu modulu rychlosti. Tyto rychlosti pak pouze předá po sběrnici sekundárním modulům. Dále do počítače přepošle po sériové lince všechny odezvy od sekundárních modulů. Pro tento test je potřeba již naprogramovat a zkompletovat všechny sekundární moduly. Následně je třeba

je připojit sběrnicí CAN. Z tohoto vyplývá, že pro tento test je potřeba plná implementace sekundárních modulů.

Při testu pak kontroluji pohyb motorů a obsah příchozích zpráv na primárním modulu. V průběhu testu je pak třeba měnit žádané rychlosti motorů a pozorovat odezvu systému. V první řadě by se měly měnit stavy v rámci systému. Dále by se měla příslušným způsobem měnit rychlosti motorů. Při tomto testu je možné zkontrolovat také chování při zátěži.

Tento test proběhl úspěšně. Motory správně reagovaly na zadané povely. Na nouzový stav byla reakce okamžitá. Zprávy o stavech motorů byli také včas zasílané primárnímu modulu.

4.5 RC ovládání

V tomto testu je třeba primárně otestovat příjem z dálkového ovládání. Toto je realizováno za použití modulu AR400[12] a dálkového ovládání DX5e [22]. Pro testování tohoto rozhraní jsem zvolil dva přístupy.

4.5.1 Příjem z ovladače

V rámci tohoto testu testuji pouze komunikaci mezi dálkovým ovládáním a primárním modulem. Tuto komunikaci zprostředkovává modul AR400 [12]. Jedná se o jednoduchý RC přijímač. Modul umožňuje komunikaci za použití čtyř kanálů. Každý z těchto kanálů přímo odpovídá pozici jedné z os páček dálkového ovládání. Výstup tohoto modulu je ve formě PWM signálu, který je třeba přečíst primárním modulem. Pro vyčtení délky PWM signálu implementuje Arduino[15] funkci *PulseIn()*[58], která umožňuje měřit délku signálu na vstupu mezi vzestupnou a sestupnou hranou.

V první fázi testu bylo třeba zjistit, jak výstup koreluje se pozicemi páček. Pro vyčtení těchto hodnot z primárního modulu je použita sériová linka. Testoval pozice páček a jejich korelace s pozicemi páček na ovladači. Dále bylo třeba ověřit detekci odpojeného ovladače. V poslední řadě bylo třeba otestovat přepočet různých pozic páček na žádané rychlosti.

V průběhu tohoto testu jsem objevil několik specifických vlastností výstupu ovladače. Na ovladači je možné si nastavit obrácení hodnot páček. Neinvertované páčky fungují následovně. Je-li signál na vertikální ose z hlediska ovladače, pak hodnota signálu roste, čím blíže je páčka k horní straně ovladače. Je-li signál na horizontální ose, pak hodnota na výstupu neinvertovaného výstupu roste, čím blíže je páčka k pravé straně ovladače. Hodnoty na ovladači je dále možné korigovat o malé hodnoty, za pomocí malých páček po stranách každého kanálu.

Během testu jsem zjistil, že kanál THR se chová jinak než ostatní kanály. Osa páčky odpovídající kanálu THR se nevrací do polohy odpovídající střední hodnotě. Ostatní kanály mají rozsah hodnot mezi 1920 a 1120 [μs] tedy rozsah 800 [μs]. Pro kanál THR se pohybuje na rozsahu 1920 a 1320 [μs] tedy

rozsah 600 [μs]. Proto k tomuto kanálu bylo třeba přistupovat jinak, než k ostatním kanálům. Z těchto vlastností mi vyšlo, že THR není úplně vhodný ke konzistentnímu použití.

Velkou částí tohoto testu je také detekce absence ovladače nebo přijímače. Je-li vysílač odpojen nebo je-li přijímač odpojen, projeví se to absencí PWM na vstupu. Detekci tohoto problému je proto třeba provést za pomocí časované verze funkce *PulseIn()*[58]. Při tomto použití je možné detekovat absenci začátku vlny. Zařízení detekuje tento stav za pomocí výstupu, této funkce.

Při provádění testu se také projevila zbytečná přesnost vyčítání vstupů z ovladače. To vedlo k mírné oscilaci hodnot. Proto je třeba snížit přesnost výsledku. Hodnoty oscilovaly okolo skutečné hodnoty. Pro stabilnější chování jsem se rozhodl snížit přesnost snímačů o 10. Toto vedlo k daleko žádanější odezvě.

V další řadě je třeba přepočítat hodnoty z páček na žádaný výkon dané strany vozítka. Toto zjednodušení je možno udělat několika způsoby. Prvním způsobem je vypočítat výkon ze zpracovaných poloh vertikálních páček ovladače. Vzhledem k tomu, že jedna z vertikálních páček odpovídá THR, je třeba pro každou stranu přepočítávat rychlost jinak. Z předešlých experimentů jsem se pro jednotlivé motory zvolil maximální rychlost 1500 v libovolném směru. Z intervalu 1920 až 1120 vznikne interval osmi set hodnot. Z intervalu 1920 až 1320 vznikne interval šesti set hodnot. Střed páčky je přesně mezi těmito hodnotami. Od tohoto středu se pak přepočítává, zda je rychlost dopředu, či dozadu. Z tohoto testu vyšlo že tento přepočet není úplně vhodný kvůli rozdílnému chování páček.

Druhou alternativou bylo zpracovat souřadnice polohy jedné z páček. Z této polohy odvodit vektor, kterým se má vozítko pohybovat. Z tohoto vektoru je pak třeba vypočítat výkony pro jednotlivé strany vozítka. Základem tohoto je devět bodů na páčce. V první řadě, je-li páčka ve středu, vozítko by mělo zůstat nehybné a výkon motorů by měl být $< 0; 0 >$ %. Je-li páčka pak na plno na horizontální ose, znamená to pohyb plnou rychlostí daným směrem a rychlosti jsou $< 100, 100 >$ %, je-li páčka na středu nahoře a $< -100, -100 >$ % je-li páčka na středu dole. Na horizontální ose funguje tento pohyb obdobně, akorát motory mají prohozená znaménka a vozítko se točí na místě. Poslední čtyři body jsou, jsou-li obě polohy maximalizovány. V tomto případě je třeba, aby vozítko zastavilo jeden z motorů a druhý běžel na plný výkon daným směrem. Body mezi těmito body jsou pak složeny z funkcí, které definují chování v daných bodech. To umožňuje plynulé ovládání za pomocí jedné páčky.

4.5.2 Ovládané vozítko

Vzhledem k tomu, že všechny předchozí testy proběhly úspěšně je již možné otestovat primitivní primární modul. Zatím nejsou otestovány senzory, takže je potřeba, aby operátor vozítka kontroloval jeho bezpečný provoz.

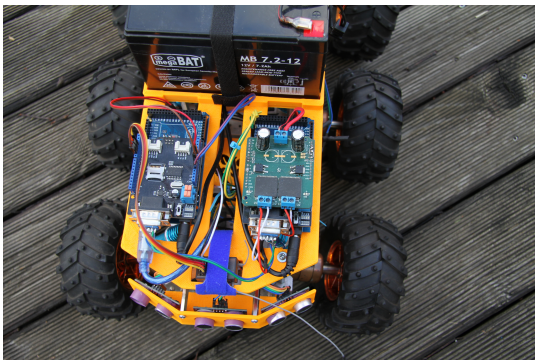
K tomuto testu je již zapotřebí akumulátor. Z akumulátoru musí být napájené všechny moduly přímo zapojené na vozítko. Poté je třeba zvolit zdroj pro 5 [V] obvod. Na vozítku jsem se toto rozhodl realizovat za pomocí primárního modulu. Při tomto testu ještě není akumulátor měřen, proto musí operátor před započetím testu zkontrolovat jeho stav.

Dále jsou potřeba, aby byly plně realizovány všechny sekundární moduly. Tyto moduly je třeba za pomocí CAN zapojit na primární modul. Na tento modul dále musí být zapojen modul AR400[12]. Na základě chování tohoto modulu musí primární modul zasílat sekundárním modulům správné příkazy.

Test probíhá následovně. Uživatel zapne vozítko a poté se pokouší mu zadat povely. Následně kontroluje, že vozítko se pohybuje dle povelů. Hlavním testovaným faktorem je odezva vozítka na povely z ovladače. Je třeba aby odezva byla co nejrychlejší. Během testu je také možno zkontrolovat odezvu celého systému na odpojení ovladače. V tomto případě se očekává okamžité zastavení vozítka.

Uvedený test je nezbytný pro další funkce primárního modulu. Je to jeden ze základních způsobů ovládání. Vozítko reagovalo dle očekávání. Během tohoto testu také již bylo zřejmé, že používání dvou páček k ovládání vozítka, když se chovají rozdílně, je uživatelsky nepřívětivé.

Dále se při tomto testu se ukázaly další nedostatky L298N. Tento modul, kromě jiných vad, omezuje výkon, který mohou motory využít. Toto omezení způsobí, že do motorů nejde dostatečný výkon. Modul stačí k ovládání motorů bez zátěže, nicméně je nemožné pří jeho využití s vozítkem zatáčet. To vedlo k nutnosti použití VNH3SP30[7].



Obrázek 4.2: Předek vozítka a propojené moduly

4.6 Senzory

Na podvozku jsou použity dva typy senzorů. Každý z nich monitoruje jiné parametry vozítka.

4.6.1 Měření akumulátoru

Pro měření akumulátoru je využit jednoduchý obvod. Tento obvod je odporová dělička. Tento obvod je použit, protože rozhodovací hodnota Arduina je 5 [V]. Akumulátor umožňuje napájení 12 [V]. Vzhledem k tomu, jak je tento akumulátor postaven, tato hodnota není přesná. Napětí také klesá, jak se akumulátor vybíjí. Proto je třeba tento pokles monitorovat.

Pohybuje-li se napětí akumulátoru nad hranicí 10 [V], pak je provoz vozítka bezpečný. Jak jsem se zmínil akumulátor může mít napětí vyšší, než 12 [V]. Z tohoto důvodu je odporová dělička dimenzována až na 20 [V]. To umožňuje dostatečně přesné měření akumulátoru při zachování bezpečného napětí na zdroji.

Nejprve je třeba otestovat funkci odporové děličky. Pro tento test je třeba s dostatečnou přesností měnit napětí na vstupu obvodu. K tomuto účelu jsem použil laboratorní zdroje[57]. Při změně vstupu obvodu je třeba monitorovat výstup obvodu. K tomuto účelu lze použít Arduino nebo libovolný voltmetr. Výstup tohoto obvodu by pak měl odpovídat poměrně vstupu. Test pak probíhá způsobem, že na vstupu obvodu mění uživatel vstup a kontroluje změnu výstupu.

Dále je třeba změřit vliv vozítka na napětí vyčtené z akumulátoru. Při provádění tohoto testu jde hlavně o kalibraci měřícího zařízení. Hlavní zkoumanou veličinou je rozdíl mezi výstupním napětím nezatíženého akumulátoru a zatíženého. Očekávaným výstupem nebyl velký rozdíl, mezi těmito výsledky. Je-li tento rozdíl zanedbatelný, pak výstup tohoto zařízení je bezezbytku použitelný pro monitorování akumulátoru v celém prototypu.

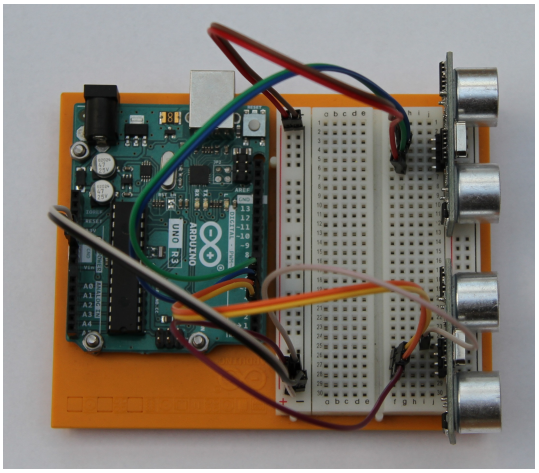
Testy proběhly úspěšně. Rozdíl v napětí odpovídal cílenému poměru a výstup obvodu zůstával v rámci přiměřené chyby i po připojení zátěže. Toto monitorování je proto vhodné k použití v konečném prototypu.

4.6.2 Měření vzdálenosti

Pro orientaci v prostoru je třeba vnímat relevantní informace v daných směrech. K tomuto jsem se v tomto projektu rozhodl využít senzory HY-SRF05[14]. Jedná se o ultrazvukový snímač vzdálenosti. Pro první užití bylo třeba otestovat vyčítání vzdálenosti ze samostatného senzoru. K tomuto vyčítání je třeba senzor napojit na samostatný Arduino modul. Následně je třeba změřit vzdálenost od překážky. Modul vrací vzdálenost ve výše zmíněné formě, proto je třeba použít i výše zmíněný výpočet.

Výsledek v této fázi je třeba vyčítat na sériové lince. Očekávané chování je, že vyčtená vzdálenost bude odpovídat reálné vzdálenosti překážky od senzoru. Dále je vhodné ověřit, že senzor vnímá v předpokládaném úhlu.

Z tohoto testu vyšlo několik poznatků. V první řadě je třeba periodicky rychle vyčítat vzdálenost ze senzorů. Dalším poznatkem je, že není možné rozpoznat, je-li překážka dále než 5 [m], blíže než 2 [cm] nebo modul je odpojen.



Obrázek 4.3: Test senzorů vzdálenosti

Rozdíl mezi přímo odpojeným modulem a modulem, který má překážku mimo vzdálenosti, které není schopen přijímat.

Na primárním modulu bude zapojeno větší množství senzorů. Proto je třeba otestovat toto zapojení na modulu. V tomto případě je třeba na modul zapojit toto množství senzorů. Následně je třeba zkoumat odezvu z nich. Vyčtení informací z těchto modulů musí probíhat pravidelně. Odezva musí být dostatečně rychlá.

Pro svůj model jsem se rozhodl využít šestici senzorů. To mi umožňuje rozšířit vnímaný úhel.

Při tomto testu se ukázalo, že odezva těchto senzorů trvá 60 [μs]. Při pravidelném vyčítání z těchto senzorů je nicméně možné jejich další užití.

Následně bylo třeba provést test na vozítka jako celku. U vozítka jako celku je třeba otestovat rychlost odezvy. Následně je třeba nakalibrovat rychlosti, kterých bude vozítko dosahovat. Při provozu vozítka je třeba předpokládat, že uživatel nemá najednou přístup ke všem informacím, nebo je nestíhá zpracovávat.

Pro tento test je třeba již třeba realizovat všechny sekundární moduly a většinu primárního modulu. Primární modul musí zpracovávat informace, ze všech zdrojů. Mezi zdroji jsou jednak senzory, přijímač vysílače a všechny sekundární moduly.

V průběhu testu dále posíláme vozítku instrukce z ovladače. Vozítko musí zastavit v čas před nárazem do detekovaných překážek. Senzory fungují do vzdálenosti 2 [cm], proto je třeba, aby vozítko zastavilo dřív, než se v daném směru dostane moc blízko překážce.

Z tohoto testu vyšlo několik poznatků. V první řadě, blíží-li se vozítko k překážce, je v daném směru je třeba omezit rychlost v dostatečné vzdálenosti od překážky. Není-li detekována překážka, pak se vozítko může hýbat plnou

rychlostí. Při absenci senzorů, nebo jejich poruše, je možné hýbat s vozítkem bezpečnou rychlostí.

4.7 Nadřazený modul

Mezi primárním a nadřazeným modulem probíhá komunikace pomocí sériové linky. Nadřazený modul vyčítat

Primární modul za tímto účelem, udržuje informace o všech jemu dostupných informacích. Pro tento test je nejprve třeba udělat simulaci těchto dat a poslat je nadřazenému modulu. Nadřazený modul pak musí tyto informace přijmout. Následně na jejich základě by měl být schopen vyvodit nové povely pro primární modul.

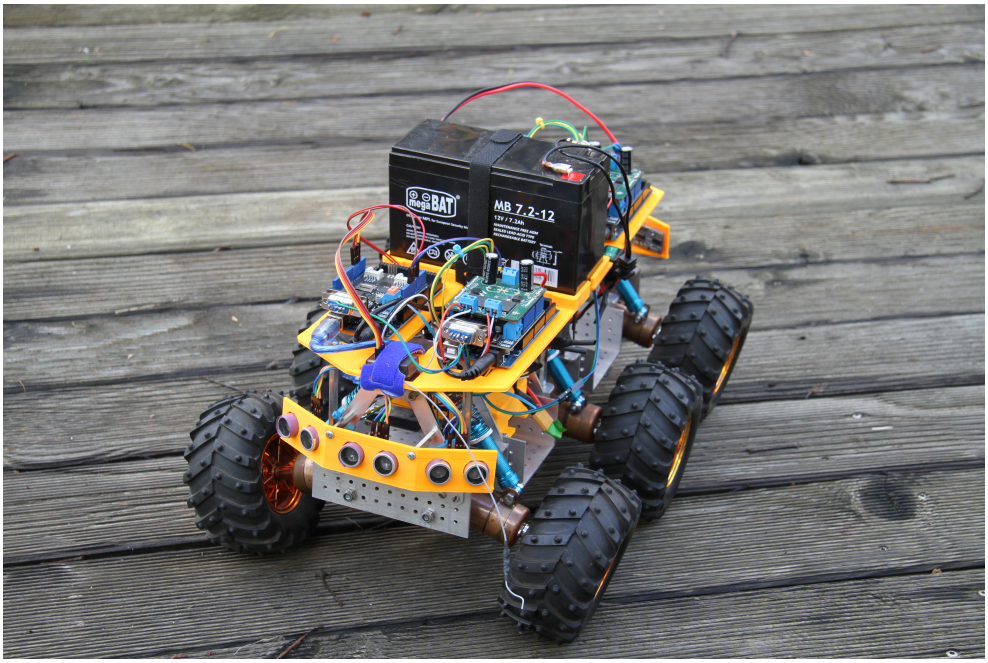
Pro tento test je třeba použít pouze Raspberry pi a Arduino. Pro napájení Raspberry pi jsem plánoval využít 5 [V] výstup Arduina. Proto jsem chtěl otestovat, že toto bude fungovat. Následně je třeba propojit sériovou linku Arduina a Raspberry pi. V průběhu testu se ukázalo, že výstup Arduina nestačí k napájení Raspberry pi nestačí. Tudíž bylo třeba přidat zdroj, který může napájet Raspberry pi a při tom být zapojen na 12 [V] zdroj. Moduly je pak třeba propojit za použití USB kabelu. Z Raspberry pi je pak možné detekovat připojený primární modul. Také je možné mu poslat příkazy a vyčíst z něj stav za použití sériové linky.

Tento test proběhl úspěšně. Podařilo se mi navázat komunikaci a vyčíst požadované informace a zadat možné povely.

4.8 Celé vozítko

Celé vozítko se skládá ze všech výše otestovaných modulů. Všechny moduly je třeba umístit na vozítko a zapojit je. V rámci testu je očekávané, že vozítko bude reagovat, dle povelů z ovladače či povelů z nadřazeného modulu. Také je očekávané, že nasměruje-li uživatel vozítko do detekované překážky, vozítko modifikuje příkaz tak aby nedošlo k jeho poškození. Dojde-li k odpojení některého z modulů vozítko by se mělo zastavit

Vozítko se při testu chovalo dle očekávání. Z tohoto testu nevyšli již žádné změny.



Obrázek 4.4: Celé vozítko

Závěr

V rámci práce jsem našel existující řešení a zvážil jejich vlastnosti. Na základě těchto řešení jsem pak navrhl vlastní systém.

Tento systém se skládá z primárního a sekundárních modulů. Každý sekundární modul ovládá dvojici kol. Systém umožňuje řízení vozítka všemi směry a detekovat překážky. K přijmu pokynů od uživatele systém používá RC přijímač. V závislosti na okolí a stavu komponent umí systém modifikovat přijaté příkazy k zajištění bezpečného chodu vozítka.

V rámci projektu jsem sestavil prototyp tohoto systému. Prototyp je realizován za využití platformy Arduino. Vzniklé vozítko je napájené z akumulátoru. Prototyp počítá z rozšířením o modul Raspberry pi k rozšíření funkcí.

Tento prototyp jsem otestoval. Použité testy testují relevantní funkce pro provoz vozítka.

Literatura

[1] Horák, B. R.: Mobilní robot schopný pohybu ve venkovním prostředí. *Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií*, 2014.

[2] silver_a: Remote Controlled 6WD All Terrain Robot. [online], 2022 [cit. 2022-12-12]. Dostupné z: `https://www.instructables.com/Remote-Controlled-6WD-All-Terrain-Robot/`

[3] Yahboom: 6WD-Arduino-robot-car. [online], 2021 [pub. 2021-1-10]. Dostupné z: `https://github.com/YahboomTechnology/6WD-Arduino-robot-car`

[4] Zemánek, M.: Dálkově ovládané čtyřkolové vozítko založené na platformě Arduino. *Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií*, 2019.

[5] Horák, B. R.: Mobilní robot schopný pohybu ve venkovním prostředí. *Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií*, 2014: str. 59.

[6] L298N Dual H Bridge DC Motor Driver Module. *HandsOn Tech*, [online], 2018 [cit. 2022-11-21]. Dostupné z: `https://handsontec.com/index.php/product/l298n-dual-h-bridge-dc-motor-driver-module/`

[7] STM32 Dynamic Efficiency MCU, Arm Cortex-M4 core with DSP and FPU, up to 512 Kbytes of Flash memory, 84 MHz CPU, Art Accelerator. *STMicroelectronics*, [online], 2022 [cit. 2022-10-17]. Dostupné z: `https://www.st.com/en/automotive-analog-and-power/vnh3sp30-e.html`

[8] CAN-BUS Shield V2.0. *SeeedStudio*, [online], 2008 [cit. 2022-11-24]. Dostupné z: `https://wiki.seeedstudio.com/CAN-BUS_Shield_V2.0/`

[9] Arduino Uno Rev. 3. *Arduino*, [online], 2022 [cit. 2022-11-21]. Dostupné z: https://docs.arduino.cc/hardware/uno-rev3?queryID=6d26c9c3b8ddfe2bbde5d0eda2d2b5f3

[10] Arduino Mega 2560 Rev. 3. *Arduino*, [online], 2022 [cit. 2022-11-21]. Dostupné z: https://docs.arduino.cc/hardware/mega-2560?queryID=undefined

[11] Raspberry Pi 3 Model B. [online], 2022 [cit. 2022-11-21]. Dostupné z: https://www.raspberrypi.com/products/raspberry-pi-3-model-b/

[12] AR400 4-Channel DSMX Aircraft Receiver. *Spektrum*, [online], 2022 [cit. 2022-11-21]. Dostupné z: https://www.spektrumrc.com/product/ar400-4-channel-dsmx-aircraft-receiver/SPMAR400.html

[13] OpenSCAD: OpenSCAD. [online], 2018 [cit. 2022-11-26]. Dostupné z: https://www.openscad.org

[14] HY-SRF05 Precision Ultrasonic Sensor. [online], [cit. 2022-11-24]. Dostupné z: https://wiki.hshl.de/wiki/images/2/23/HY-SRF05-ETC.pdf

[15] Arduino. *Arduino*, [online], 2022 [cit. 2022-10-17]. Dostupné z: https://www.arduino.cc/

[16] CAN bus. [online], 2022 [pub. 2022-11-25]. Dostupné z: https://en.wikipedia.org/wiki/CAN_bus

[17] Raspberry Pi. [online], 2022 [cit. 2022-11-21]. Dostupné z: https://www.raspberrypi.com/

[18] STM32 Nucleo-64 development board with STM32F401RE MCU, supports Arduino and ST morpho connectivity. *STMicroelectronics*, [online], 2022 [cit. 2022-10-17]. Dostupné z: https://www.st.com/en/evaluation-tools/nucleo-f401re.html

[19] STM32 Dynamic Efficiency MCU, Arm Cortex-M4 core with DSP and FPU, up to 512 Kbytes of Flash memory, 84 MHz CPU, Art Accelerator. *STMicroelectronics*, [online], 2022 [cit. 2022-10-17]. Dostupné z: https://www.st.com/en/microcontrollers-microprocessors/stm32f401re.html

[20] Serial Peripheral Interface. [online], 2022 [pub. 2022-11-25]. Dostupné z: https://en.wikipedia.org/wiki/Serial_Peripheral_Interface

[21] USART. [online], 2021 [pub. 2021-3-29]. Dostupné z: https://cs.wikipedia.org/wiki/USART

[22] DX5e. *SPEKTRUM*, [online], 2022 [cit. 2022-10-17]. Dostupné z: https://www.spektrumrc.com/product/dx5e-5-channel-dsmx-transmitter-with-ar600-receiver-mode-2/SPM5510.html

[23] Ultrasonic Ranging Module HC - SR04. *ELECFreaks*, [online], [cit. 2022-11-24]. Dostupné z: https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf

[24] Sharp GP2Y0A21YK0F Analog Distance Sensor 10-80cm. *Pololu*, [online], [cit. 2022-11-26]. Dostupné z: https://www.pololu.com/product/136

[25] Horák, B. R.: Mobilní robot schopný pohybu ve venkovním prostředí. *Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií*, 2014: str. 28.

[26] sensirion: Čidlo vlhkosti a teploty, SMD SHT15. [online], 2014 [pub. 2004-07]. Dostupné z: https://www.gme.cz/sht15

[27] AUTOMOTIVE FULLY INTEGRATED H-BRIDGE MOTOR DRIVER. *STMicroelectronics*, [online], 2022 [cit. 2022-10-17]. Dostupné z: https://www.st.com/en/automotive-analog-and-power/vnh2sp30-e.html

[28] Atmel ATmega640/V-1280/V-1281/V-2560/V-2561/V. *Atmel*, [online], 2014 [cit. 2022-11-24]. Dostupné z: https://ww1.microchip.com/downloads/en/devicedoc/atmel-2549-8-bit-avr-microcontroller-atmega640-1280-1281-2560-2561_datasheet.pdf

[29] MK610 Receiver. *Aliexpress*, [online], 2019 [cit. 2022-11-21]. Dostupné z: https://www.aliexpress.com/item/1pcs-MK610-Receiver-2-4GHz-Replace-AR6100-Receiver-for-Dx5e-Dx6i-Dx7-Dx8/32970328829.html?+spm=a2g0s.9042311.0.0.13554c4dYeLlm

[30] Horák, B. R.: Mobilní robot schopný pohybu ve venkovním prostředí. *Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií*, 2014: str. 6.

[31] Horák, B. R.: Mobilní robot schopný pohybu ve venkovním prostředí. *Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií*, 2014: str. 6.

[32] RS232. [online], 2022 [pub. 2022-11-21]. Dostupné z: https://cs.wikipedia.org/wiki/RS-232

[33] 34:1 Metal Gearmotor 25Dx64L mm HP 6V with 48 CPR Encoder (No End Cap). *Pololu*, [online], [cit. 2022-11-26]. Dostupné z: https://www.pololu.com/product/2273

[34] 34:1 Metal Gearmotor 25Dx67L mm HP 6V with 48 CPR Encode. *Pololu*, [online], [cit. 2022-11-26]. Dostupné z: https://www.pololu.com/product/4804

[35] DC 6V 12V 24V Encoder Motor Mini Metal Gear Motor High Torque Electric Gear Reducer Reduction Motor 12-1360 RPM Geared Motor. *Aliexpress*, [online], [cit. 2022-11-26]. Dostupné z: https://www.aliexpress.com/item/1005003160300513.html?spm=a2g0o.order_list.0.0.6b2a1802Avtyoe

[36] MCP2551. *Microchip*, [online], 2001 [cit. 2022-11-24]. Dostupné z: https://www.microchip.com/en-us/product/MCP2551#document-table

[37] I2C CAN BUS MODULE. *Logan Labs*, [online], 2017 [cit. 2022-11-24]. Dostupné z: http://docs.longan-labs.cc/1030017/

[38] ATmega328P. *Atmel*, [online], 2015 [cit. 2022-11-26]. Dostupné z: https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf

[39] Arduino DUE. *Arduino*, [online], 2022 [cit. 2022-11-21]. Dostupné z: https://docs.arduino.cc/hardware/due?queryID=7af080ce88c4929704e31c0467e78887

[40] SAM3X / SAM3A Series. *Atmel*, [online], 2015 [cit. 2022-11-26]. Dostupné z: https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-11057-32-bit-Cortex-M3-Microcontroller-SAM3X-SAM3A_Datasheet.pdf

[41] Horák, B. R.: Mobilní robot schopný pohybu ve venkovním prostředí. *Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií*, 2014: str. 16.

[42] Gelová baterie 12V 7,2Ah R-AKU.12V-7AH. [online], [cit. 2022-11-26]. Dostupné z: https://www.ministerstvohracek.cz/baterie-do-elektrickeho-auticko-12v-7ah/

[43] M-Style 2x USB-A zásuvka - nabíječka 36W QC 3.0. [online], [cit. 2022-11-26]. Dostupné z: https://www.alza.cz/auto/m-style-2x-usb-a-zasuvka-nabijecka-36w-qc-3-0-d7532822.htm

[44] AR600 6-Channel Sport DSMX Receiver. *Spektrum*, [online], 2022 [cit. 2022-11-21]. Dostupné z: https://www.spektrumrc.com/product/ar600-6-channel-sport-dsmx-receiver/SPMAR600.html

[45] Eclipse: Eclipse. [online], 2022 [cit. 2022-11-26]. Dostupné z: https://www.eclipse.org/

[46] Microchip: MICROCHIP STUDIO FOR AVR® AND SAM DEVICES. [online], 2022 [cit. 2022-11-26]. Dostupné z: https://www.microchip.com/en-us/development-tool/microchip-studio#Documentation

[47] Arduino IDE. *Arduino*, [online], 2022 [cit. 2022-11-21]. Dostupné z: https://www.arduino.cc/en/software

[48] Blender. *Blender*, [online], 2022 [cit. 2022-11-26]. Dostupné z: https://www.blender.org/

[49] G-code. *Reprap.org*, [online], 2022 [cit. 2022-11-26]. Dostupné z: https://reprap.org/wiki/G-code

[50] Ranellucci, A.: Slic3r. [online], 2019 [cit. 2019-05-13]. Dostupné z: https://slic3r.org/

[51] PrusaSlicer 2.5.0. *prusa3d*, [online], 2022 [cit. 2022-11-26]. Dostupné z: https://www.prusa3d.com/page/prusaslicer_424/

[52] Utilmaker: Ultimaker Cura 4.3. [online], 2011 [cit. 2022-11-26]. Dostupné z: https://ultimaker.com/learn/ultimaker-cura-4-3-available-now

[53] USB. [online], 2022 [pub. 2022-10-26]. Dostupné z: https://cs.wikipedia.org/wiki/USB

[54] Measuring DC Voltage using Arduino. *StartingElectronics*, [online], 2013 [pub. 2013-5-23]. Dostupné z: https://startingelectronics.org/articles/arduino/measuring-voltage-with-arduino/

[55] MCP2515. *Microchip*, [online], 2003 [cit. 2022-11-24]. Dostupné z: https://www.microchip.com/en-us/product/MCP2515

[56] Nicholas_N: Distance Measurement with an Ultrasonic Sensor HY-SRF05. *Arduino*, [online], 2017 [pub. 2017-7-23]. Dostupné z: https://create.arduino.cc/projecthub/Nicholas_N/distance-measurement-with-an-ultrasonic-sensor-hy-srf05-64554e

[57] Laboratorní zdroj 0-30V/0-2,5A EP-613. [online], [cit. 2022-11-26]. Dostupné z: https://www.gme.cz/laboratorni-zdroj-manson-ep-613

[58] pulseIn(). *Arduino*. Dostupné z: https://www.arduino.cc/reference/en/language/functions/advanced-io/pulsein/

Příloha **A**

Seznam použitých zkratek

PWM Pulse-width modification

CAN Conterol area network

IR Infra red

SPI Serial peripherial interface

AD Analog to digital

DC Dirrect current

IDE Integrated development enviroment

USB Universal serial bus

HW Hard ware

RC Remote control

Příloha B

Obsah přiloženého média

- `readme.txt` ..... stručný popis obsahu média
- `src`
  - `deploy` ..... finální kódy
    - `primary` ..... Arduino IDE projekt určený pro primární modul
    - `secondary` ... Arduino IDE projekt určený pro sekundární modul
  - `test` ..... Dílčí kódy použité při tvorbě
    - `CAN_test` ..... Kódy pro testování CAN sběrnice
      - `CAN_test_primary` ..... Testovací primární modul na sběrnici
      - `CAN_test_secondary` .. Testovací sekundární modul na sběrnici
    - `encoder_with_motor` ..... Test H-můstku s enkodérem
    - `encoders_read` ..... čtení z enkodéru
    - `reciever_test` ..... čtení přijmu z AR400 a jejich přepočtů
    - `resistor_devider_test` ..... Test odporové děličky
    - `serial_motor_enc` ..... H-můstek s enkodérem a zpětnovazebného systému ovládaný sériovou linkou
    - `srf05_test` ..... Čtení ze dvou modulů HY-SRF05
  - `scad` ..... 3D modely vytvořené při práci ve formátu scad
    - `Back_Mount_top.scad` .... Model upevnění sekundárních modulů v zadní části vozítka
    - `Front_Mount_top.scad` Model upevnění primárního a sekundárního modulů v přední části vozítka
    - `srf05-mount.scad` ..... Model upevnění modulů HY-SRF05
    - `common.scad` ..... Společné definice pro ostatná soubory
  - `zemanm30.zip` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `zemanm30.pdf` ..... text práce ve formátu PDF